HZ BOOKS
华章教育

计 算 机 科 学 丛 书

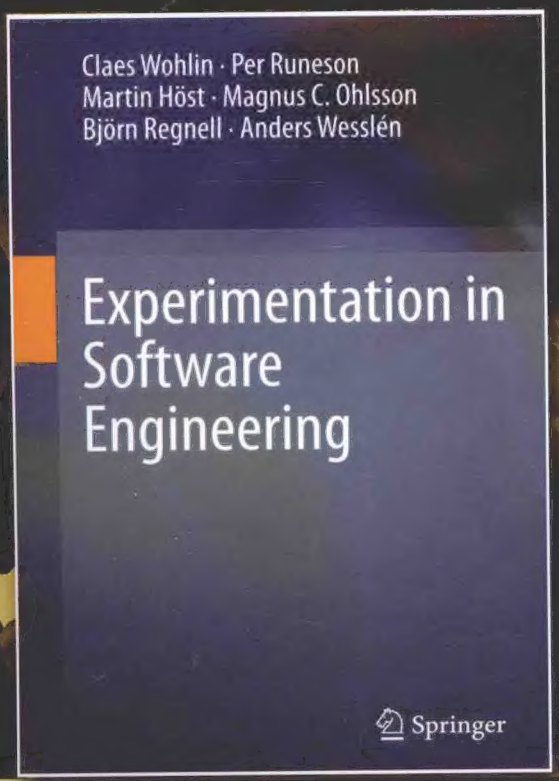
Springer

经验软件工程

软件工程中的实验研究方法

克拉斯·沃林（Claes Wohlin）
佩尔·鲁内松（Per Runeson）
马丁·霍斯特（Martin Höst）
[瑞典] 马格纳斯力 C. 欧尔松（Magnus C. Ohlsson） 著 张莉 王青 彭蓉 宣琦 译
比约恩·雷格尔（Björn Regnell）
安德斯·韦斯伦（Anders Wesslén）

Experimentation in Software Engineering



机械工业出版社
China Machine Press

经验软件工程 软件工程中的实验研究方法

Experimentation in Software Engineering

首部以描述如何系统地进行和评估软件工程实验为主要内容的著作。
书中不仅包括对实验及其步骤的完整讲述，还包含对案例研究及系统文献综述的详细介绍。

“这次改版包含了新的章节和例子，更加巩固了其作为软件工程领域设计、构造、执行和评估实验的首本著作的地位。本次改版将使本书更加有价值。它将成为所有博士研究生必读的书籍，每位学者的书架上都应该备上一本以备查阅。”

Michael Oudshoorn, Computing Reviews, 2012年10月

“这本书是一个里程碑，它使得我们能够依此训练从事软件工程实验的研究者和从业者。”

Victor R. Basili, 马里兰大学

“改版中新增的和修改的部分非常好地反映了经验软件工程这一领域的成熟。”

Anneliese A. Andrews, 丹佛大学

“本次改版与2000年出版的著作同名，详细阐述了软件工程研究中的各种方法。它以教科书的方式呈现，使得其非常适合作为研究生的导论课程或者本科生四年级课程的教材。从业者和专家也能将此书作为使用更加深奥的方法的起点而从本书受益……总而言之：非常推荐从大学三年级以上学生到资深研究者/从业者使用。”

L. Benedicenti, Choice, 第50卷第9期，2013年5月

内容介绍

与其他科学与工程学科相似，软件工程需要一个建模、实验和学习的循环。在评价及选择不同的方法、技术、语言和工具时，实验对于所有的软件工程师而言都是非常有价值的工具。

本书的目的是通过受控实验为学生、教师、研究人员及从业者介绍软件工程中的经验研究。介绍实验时采用了过程视角，将描述的焦点放在进行实验时需要执行的步骤上。全书分为三个部分：第一部分介绍了实验中用到的一些理论和方法的背景知识；第二部分包括五个章节，分别介绍了实验的五个步骤：确定范围、计划、操作、分析与结果展示。第三部分完整展示了两个案例。附录中提供了课后作业与统计方面的资料。就整体而言，本书不仅为经验型研究（特别是实验）提供了不可或缺的信息，而且还介绍了案例研究、系统文献综述和问卷调查等方法。本书是作者们2000年出版书的修改版。另外，还增加了大量的新内容，如关于系统文献综述及案例研究的介绍。

本书是自包含的，能够作为需要学习软件工程经验研究的本科生或研究生的教材。提供的练习及作业能够促进理论与实践相结合。研究人员也能从本书中获得收益，学到如何进行经验型研究；同时从业者也可以在其公司引入新的方法或技术时将之作为指导如何评价这些方法和技术的作业指南。

Springer

投稿热线：(010) 88379604
客服热线：(010) 88378991 88361066
购书热线：(010) 68326294 88379649 68995259

华章网站：www.hzbook.com
网上购书：www.china-pub.com
数字阅读：www.hzmedia.com.cn

上架指导：计算机\软件工程

ISBN 978-7-111-51856-3

9 787111 518563 >

定价：59.00元

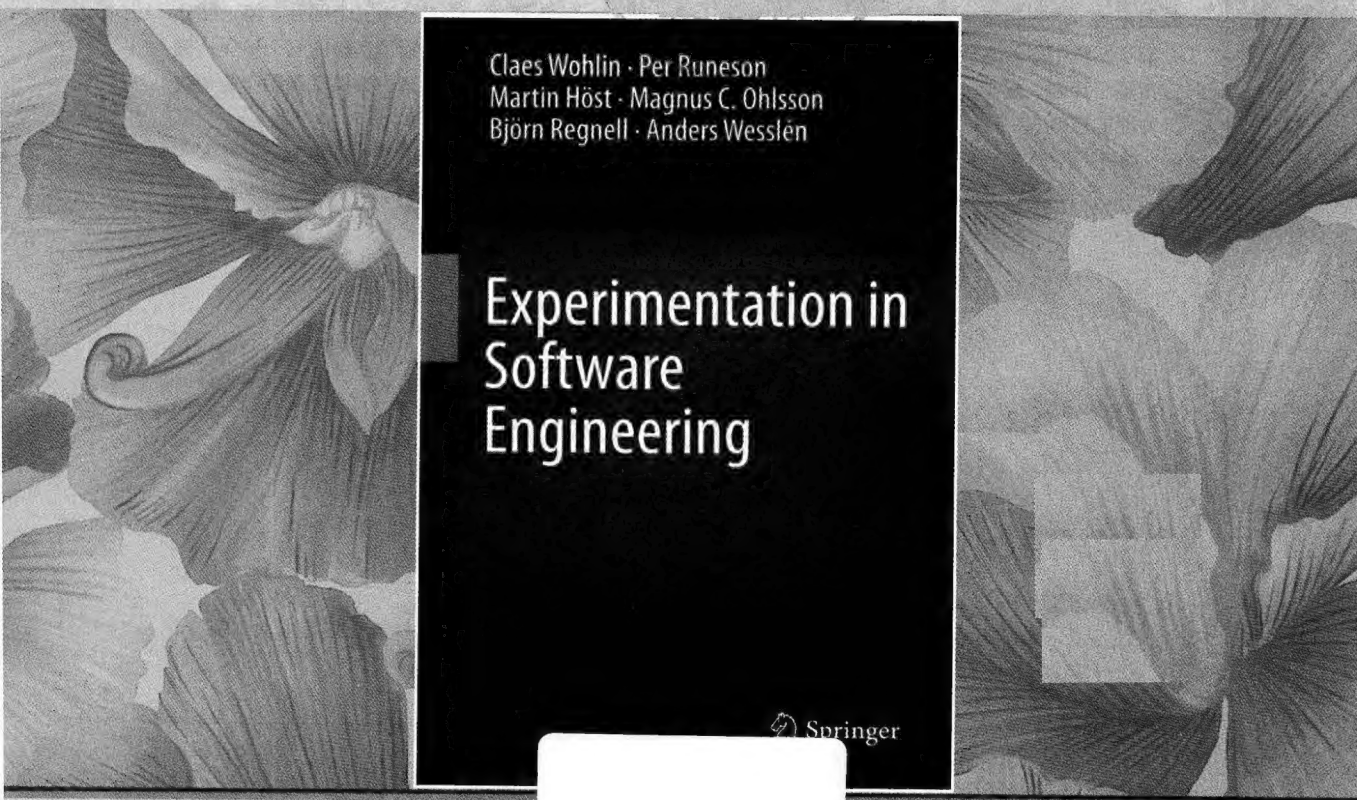
计 算 机 科 学 丛

经验软件工程

软件工程中的实验研究方法

[瑞典] 克拉斯·沃林（Claes Wohlin）
佩尔·鲁内松（Per Runeson）
马丁·霍斯特（Martin Höst）
马格纳斯力 C．欧尔松（Magnus C. Ohlsson）
比约恩·雷格尔（Björn Regnell）
安德斯·韦斯伦（Anders Wesslén）
著 张莉 王青 彭蓉 宣琦 译

Experimentation in Software Engineering



机械工业出版社
China Machine Press

图书在版编目（CIP）数据

经验软件工程：软件工程中的实验研究方法 /（瑞典）沃林（Wohlin, C.）等著；张莉等译．
—北京：机械工业出版社，2015.10
（计算机科学丛书）
书名原文：Experimentation in Software Engineering

ISBN 978-7-111-51856-3

I. 经… II. ① 沃… ② 张… III. 软件工程－研究方法 IV. TP311.5

中国版本图书馆 CIP 数据核字（2015）第 242587 号

本书版权登记号：图字：01-2015-1280

本书的目的是通过受控实验为学生、教师、研究人员及从业者介绍软件工程中的经验研究。介绍实验时采用了过程视角，将描述的焦点放在进行实验时需要执行的步骤上。全书分为三个部分。第一部分介绍了实验中用到的一些理论和方法的背景知识；第二部分包括 5 章，分别介绍了实验的五个步骤：确定范围、计划、操作、分析与结果展示。第三部分完整地展示了两个案例。附录中提供了课后作业与统计方面的资料。

本书适合作为软件工程专业高年级本科生、研究生教材，也可供企业工程技术人员使用。

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）
责任编辑：关 敏 曲 熠 姚 蕾 责任校对：殷 虹
印 刷：北京诚信伟业印刷有限公司 版 次：2015 年 10 月第 1 版第 1 次印刷
开 本：185mm×260mm 1/16 印 张：13
书 号：ISBN 978-7-111-51856-3 定 价：59.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：(010) 88378991 88361066 投稿热线：(010) 88379604
购书热线：(010) 68326294 88379649 68995259 读者信箱：hzjsj@hzbook.com

文艺复兴以来，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的优势，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭示了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短的现状下，美国等发达国家在其计算机科学发展的几十年间积淀和发展的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起到积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章公司较早意识到“出版要为教育服务”。自 1998 年开始，我们就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与 Pearson，McGraw-Hill，Elsevier，MIT，John Wiley & Sons，Cengage 等世界著名出版公司建立了良好的合作关系，从他们现有的数百种教材中甄选出 Andrew S. Tanenbaum，Bjarne Stroustrup，Brain W. Kernighan，Dennis Ritchie，Jim Gray，Afred V. Aho，John E. Hopcroft，Jeffrey D. Ullman，Abraham Silberschatz，William Stallings，Donald E. Knuth，John L. Hennessy，Larry L. Peterson 等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及珍藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力相助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专门为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近两百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍。其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证。随着计算机科学与技术专业学科建设的不断完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都将步入一个新的阶段，我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

华章网站：www.hzbook.com
电子邮件：hzjsj@hzbook.com
联系电话：（010）88379604
联系地址：北京市西城区百万庄南街 1 号
邮政编码：100037

华章科技图书出版中心

中文版序

Experimentation in Software Engineering

It is a sincere pleasure to write this foreword to the Chinese version of our book "Experimentation in Software Engineering". We never imagined that the book would be translated, and we are deeply honoured.

能够为《Experimentation in Software Engineering》这本书的中文版写序，我们感到由衷的高兴。我们从没想过这本书会被翻译成中文，对此我们深感荣幸。

The whole book project started as a PhD course in the late 1990s. The course was based on a number of articles, since it was impossible to find a course book at that time. As assignments in the course, each PhD student was asked to summarize a step in the experimentation process that we defined based on the available literature. The hand-ins formed the basis for several of the chapters, which later was extended and complemented with other parts needed to make a comprehensive book on the topic.

本书的创作开始于上世纪90年代末的一门博士课程。由于那时无法为这门课程找到合适的教科书，我们必须以很多论文为基础。作为课程作业，要求每个博士生基于当时的文献，总结并定义实验过程中的一个步骤。这些交上来的作业成为本书多个章节的基础，后来我们又对其他需要的部分进行扩展和补充，使其成为该主题一本全面的书。

The first edition was published in the end of year 2000, and we then got the opportunity to release a second edition in 2012. The field had matured since the first edition, and hence it felt very good to be able to make additions as well as revising some parts of the first edition. We perceive that the book has been well received both by students and fellow researchers in empirical software engineering, which is very rewarding based on all work put in to turn our material into a book. We hope that the Chinese edition helps us reaching out to an even broader readership, and that it can help inspiring more research in empirical software engineering.

第一版于2000年年末出版，在2012年我们得到了出版第二版的机会。自第一版出版后该领域已臻成熟，因此，我们能够很好地对第一版进行扩展和修订。我们已经可以看到，这本书在经验软件工程领域的学生和研究者中广受欢迎，我们把当初的材料整合成本书所付出的努力是非常值得的。我们希望中文版能够让本书读者群更加广泛，也希望能够鼓舞更多的研究者投入到经验软件工程领域中来。

I would like to express my sincere thanks to my former PhD students (three of them now holding full professor positions at Lund university, Sweden and two of them working with soft-

ware-intensive systems in industry) for the inspiring collaboration on what turned out to be a very successful book project from my point of view.

这里我也非常诚挚地感谢之前的博士生（其中有3个人已经获得了瑞典Lund大学的全职教授职位，还有两个人在工业界，从事软件密集型系统方面的工作）。在我看来，我们富有开创性的合作最终成就了这本书的成功。

Finally, I would like to extend my deepest gratitude to the initiators of publishing a Chinese edition and to the translators of the book.

最后我也想对出版本书中文版的发起者和翻译者表达最深切的感激。

Claes Wohlin

2015年6月15日

于瑞典卡尔斯克鲁纳市布京理工学院

译者序

Experimentation in Software Engineering

自软件工程成为一级学科以来，关于软件工程的研究方法得到了更多、更广泛的重视。软件工程领域日益需要一本这样的教材。

任何学科的发展都依赖于对这个学科所要解决的基本问题的理解。每个学科解决问题的能力都会随着领域经验的提高而提高。其基本方法是把经验封装到模型中，并基于实验、经验证据和反馈来验证并确认模型的正确性。对知识的封装可以让我们站在更高的抽象层次上理解我们的问题空间和解决空间，并通过应用反馈和学习知道哪些方法行之有效。这是一种在很多领域都适用的方法，譬如物理、医学、制造业等。这是一个建模、实验、学习和再建模的闭环。软件工程也是一门实验科学，我们必须从应用中学习并改进我们对软件工程世界的理解。经验软件工程（Empirical Software Engineering）就是这样一种方法。

2012 年在 Lionel Brand 教授等人的帮助下，北京航空航天大学开始在研究生中开设这门课程，后来得知很多学校都已经或者正在准备开设“经验软件工程”课程，但是都苦于没有合适的中文版教材。计算机学会（CCF）软件工程专委会成立“经验软件工程学组”时，与会老师进一步表达了这一愿望。经过仔细考虑，我们选择了 Claes Wohlin 等撰写的《Experimentation in Software Engineering》这本教材，并由来自北京航空航天大学的张莉、中国科学院软件研究所的王青、武汉大学的彭蓉、浙江工业大学的宣琦组成了翻译小组，在机械工业出版社华章公司的协助下开始了本书的翻译。

Empirical 在英文中是指“经验的”或“实证的”。因此，Empirical Software Engineering 被译为“经验软件工程”或“实证软件工程”。但在英文中 Empirical 区别于 Experimental，这里的“经验”不单单是人在实践中的主观体验和认知，更强调从实践中获得的客观证据。

《Experimentation in Software Engineering》这本书虽然重点阐述了在软件工程领域如何进行实验研究，但其中首先讨论了为什么要在软件工程领域展开经验研究，之后介绍了主要的经验研究策略、软件工程中常用的经验研究方法，因此，我们认为可以将其作为“经验软件工程”的一本很好的入门教材。建议将本书作为软件工程专业高年级本科生、研究生教材，本书也可供企业工程技术人员使用。

经过近一年的时间，翻译小组多次协调讨论，并就一些问题和原书作者进行了沟通，最终完成了翻译工作。由于这是国内第一本关于经验软件工程的书籍，所以翻译中最大的挑战是术语的翻译。虽然经多次推敲，也难免有不当之处，希望得到大家的指正。

译者

2015 年 9 月 10 日

实验是任何科学与工程研究的基础。

了解一门学科需要建立与该学科各种元素相关的模型，比如领域中的对象和用于操作这些对象的流程，以及流程与对象之间的关系。领域知识的不断发展意味着需通过各种形式的实验来检验演化模型。对实验结果的分析涉及学习、对知识的封装，以及随时改变和精炼模型的能力。基于此，我们对一门学科的理解也会随着时间的推移而不断地加深。

各种范式已经广泛应用于诸多领域，比如物理、医学和制造业等领域。当这些领域开始使用建模－实验－学习循环模式时，它们才逐渐发展成了独立的学科。每个领域都从记录观察结果开始，逐渐演化为调节模型变量并研究变量改变所导致的结果。各个领域在本质上不尽相同，体现在构成领域的基本对象、对象的基本特性、包含这些对象的系统的基本性质、系统中对象和系统本身以及学科文化之间的关系等方面。这些差异都会影响系统建模和实验执行。

与其他学科一样，软件工程也需要建模－实验－学习这种循环模式。软件工程研究是一门实验科学。从事该学科的人主要可分为研究者和从业者。研究者的职责是了解对象（产品）的性质，了解创建和操作这些对象的过程，以及了解系统范畴内两者之间的关系；从业者的职责是利用现有最新的知识构建更完善的系统。这两类人员互惠共生：研究者需要在实验室研究从业者遇到的问题，利用实验来提出并改进解决方案；从业者需了解如何构建更好的系统，而关于这一点研究者可以通过构建模型来提供帮助。

在建模和实验中，研究者和从业者都需要了解软件工程学科的本质。不同的软件之间存在差异：大量的变量造成了这些差异，因此理解这些变量的作用极为重要。就像药物，其中关于人类个体遗传和药物史的差异往往是构建医药模型并解释实验结果的主要因素，软件工程会处理大量不同的能影响输入和结果的情境。在软件工程中，许多技术通常涉及人工的参与而非完全自动化。就如在制造业中，主要的问题就是理解和改进流程与其生产的产品之间的关系。然而，与制造业不同，软件工程的流程是开发而不是生产，因而我们不能收集来自于完全重复的同一个流程的数据。这也促使我们从更高的抽象水平去构建模型，尽管如此，我们仍需仔细考量上下文变量。

目前，存在的模型尚不足以让我们理解软件工程学科；对处理某些特定情况技术的局限性尚缺乏认知；分析与实验的不足仍然存在。尽管存在诸多不足，但是关于最后这一点，本书将表明情况正在逐步改观。

本书对经验软件工程研究者和从业者而言具有极为重要的参考价值，是对该领域的一大贡献，具有里程碑意义。作者收集了大量的知识并极富条理地将其撰写成书，同时提供了范围确定、计划制定、运行、分析、解释和归档等一系列实验流程。它们

涵盖了从有效性威胁到统计程序的所有必要的课题。

本书涵盖了执行软件工程实验所需的大量信息。在以前做实验的时候，我需要寻找不同的信息资源，这些信息几乎全都是从其他学科获取的，我需要尽我所能将其用于满足自己的需求。如果在那时候我能拥有这本书，它必将节约我大量时间和精力，而且极有可能使我的实验变得更好！

Victor R. Basili 教授

非常荣幸能为本书的修订版作序（最初版本于2000年出版）。我之前使用过本书的最初版本，因为这本书正是为教师和研究者出版的！多年来，包括科罗拉多州立大学、华盛顿州立大学、丹佛大学和维尔茨堡综合大学等多所高校的学生来上我的课时，我均使用此书。这些学生有些是大公司的全职员工来攻读系统工程硕士学位，其他则是全日制的硕士和博士生。这本书对他们非常有帮助。除了学习书中的经验软件工程方法之外，他们也喜欢本书整体上的简洁性。我非常高兴看到修订后的版本与最初版本一样，仍然将整书的紧凑和简单放在第一位。

与最初版本相比，此版本增加和修改的内容完美地反映了经验软件工程学科当前的成熟度：包括实验重现和综合分析的日益重要，以及学术界和专业领域对基于可信定量验证的新技术成功转化这一需求的日益增强。另一个重要的进展是对于经验软件工程实验伦理的扩展讨论。特别是因为该领域没有关于实验伦理的正式定义，所以对学生而言，知道这些问题，以及能够得到一个处理这类问题的具体指导方针都是极其重要和有价值的。

本书的最初版本非常重视实验。然而在工业领域，人们更倾向于利用案例研究来评估技术、软件工程流程和软件产品。因此额外增加案例研究这一章非常有必要，也将会很受欢迎。增加系统文献综述这一章也是如此。

我使用本书的最初版本教授定量软件工程这门课程已经有十几年了，本修订版新增的内容整合了我多年来需要另外添加的课程资料。更重要的是，在增加这些新内容的同时，它并没有丢失最初版本的紧凑和简洁性。对我个人而言，我很开心能有此修订版，并会继续将它作为我上课的教程以及我学生的研究资源。

Anneliese Amschler Andrews 教授

最初版前言

Experimentation in Software Engineering

我有一种强烈的信念，即软件工程师不仅需要了解软件工程的具体方法和流程，而且应该知道如何去评估它们。因此，我一直把实验与经验研究的原则作为软件工程课程的重要组成部分。直到现在，这仍然意味着我们需要从其他学科选择教科书，通常是心理学，然后将该教科书与期刊或会议论文相结合，从而为我们的学生提供软件工程领域的实验和经验研究案例。

本书填补了软件工程书籍中的一个空白，它简洁却不失全面地审视了软件工程的一个重要方面：对软件工程的方法、方法论、流程执行的出色程度做了实验分析。由于在我们这个领域所有这些变化都非常迅速，所以知道如何评估新的方法和流程非常重要。本书指导我们该如何处理这些问题，因此本书不仅对软件工程领域的学生有用，对软件工程应用方面的专业人士也极具价值。这些专业人士能：

- 评估软件工程技术。
- 确定在已发表的论文中软件工程方法和流程的应用价值或存在的缺陷。

本书可以说是软件工程研究人员的宝贵资源。

Anneliese Amschler Andrews 教授（原名：von Mayrhauser）

你是否曾经有过对不同的软件工程方法或技术进行比较评估的需求？本书介绍了一种用实验对软件工程中的新方法和新技术进行评估的方法。实验对于那些参与评估不同方法、技术、语言和工具，并从中加以选择的所有软件工程师而言是一种极具价值的工具。

你或许是一个软件从业者，在将软件正式引进单位之前，希望能对其采用的方法和技术进行评估。你也可能是一个研究者，希望将新的研究成果与已有的研究成果进行比较，从而为新的思想提供科学基础。你或许是一个老师，认为在软件工程中基于经验学习的知识对学生来说至关重要。当然，你也可能是一个软件工程专业的学生，希望学习一些方法使软件工程成为一门科学，以及在比较不同的方法和技术时能够获得定量化的数据。本书将就如何成功实现这些目标提供相关的指导和案例。

软件工程与科学

“软件工程”这一术语诞生于1968年，该领域目前仍处于发展阶段。在过去一段时间，科技的发展和宣传式研究驱动着软件工程的发展。所谓的宣传式研究是指，我们发明和引入的新方法、新技术大都是基于市场营销和理念，而非科学的结果。在某种程度上，这也是可以理解的，因为信息社会本身也是在过去的几十年中逐步建立的。但是，从长远来看，如果我们想拥有我们所开发软件的控制权，这又是不可接受的。控制来源于我们希望在使用新的方法、技术、语言和工具之前首先对它们进行评估。而这也将有助于我们把软件工程转变成一门真正意义上的科学。在审视“我们必须把软件工程变成科学”这个问题之前，让我们先来看看其他领域是怎么看待科学的。

在西蒙·辛格（Simon Singh）博士的“费马大定理”中［160］，对科学进行了讨论，总结如下：在科学中，物理现象通常是在假说的基础上提出的。我们观察现象，如果观察到的现象与假说相一致，它将成为假说成立的证据；而另一方面，假说也应该能够帮助预测其他的现象。实验对于验证假说，尤其是对于验证假说的预测能力，是极为重要的。如果新的实验支持某一假说，则我们就有更多的证据支持这一假说。随着证据的增加和增强，假说会被广泛接受，从而成为一种新的科学理论。

科学的要点就是通过经验研究进行假设检验，然而，当前软件工程领域的大部分研究却并不是按照这个方式来进行的。尽管如此，相比于10年前，如今通过实验对新的研究方案进行评估和验证的需求却达到了更高层次的水平。经验研究通常包括调查、实验和案例研究。因此，本书的目的是介绍和促进经验研究在软件工程中的应用，其中我们尤其重视实验的方法。

本书的目的

本书主要为学生、教师、研究者和从业者介绍针对软件工程的实验方法学和经验评估方法。主要目标是提供关于在软件工程中如何进行实验来评估方法、技术和工具的指导方针，虽然其中也会穿插其他经验性方法的简短介绍。我们将从过程的角度来介绍实验，重点将关注执行实验所必须经历的操作步骤。该过程可推广到其他类型的经验研究，但本书的关注点仅限于实验和准实验。

写作本书是为了支持我们所常经历的需求：使软件工程的研究更加具有实验特色。现在已有一些相关的书籍，它们或者以十分笼统的方式来介绍该课题，或者只关注实验的某些特别的部分，且大部分都集中在实验的统计方法上。虽然这些都很重要，但目前仍然缺乏从过程角度来阐述实验的书籍。此外，我们几乎找不到介绍软件工程实验学的相关书籍。事实上，在本书最初版本出版之前的确不存在这样的书。

本书的范围

本书主要关注软件工程中的实验学，即通过实验来评估方法、技术等。本书提供了一些关于广义经验研究的信息：包括案例研究、系统文献综述和调查法。目的是对这些不同的经验研究方法做些简要的说明，使读者能够初步了解并将它们与实验相结合。

书中的各章涵盖了软件工程领域执行实验过程中不同的操作步骤。此外，关于软件工程经验研究的例子也贯穿整书。能向软件工程师说明经验研究和实验能够在软件工程中成功应用这一点对本书而言非常重要。本书包含两个实验的例子。引入这些例子是为了说明具体的实验过程以及如何将软件工程中的实验进行报告。我们的目的是使这些研究能成为未来软件工程经验研究的良好例子和灵感来源。本书注重实验，但不仅限于实验，即我们也提供其他的方法，比如案例研究和调查法。换句话说，当可以使用实验这样的研究策略时，我们就没有必要再求助于不包含定量数据的宣传研究和市场营销策略。

针对的读者

本书的目标读者大致上可以分为以下四类。

学生　可以将本书作为软件工程中侧重于评估的实验导论。本书适合作为强调经验研究的软件工程领域本科生或研究生的教科书。本书包含的习题和项目任务可以帮助读者将更多的理论素材与实际应用相结合。

教师　如果觉得软件工程课程需要引入更多的经验研究，则可以在他们的课程中使用这本书。本书适合作为该领域的入门教程。虽然我们建议同时参考统计学的导论课程，但本书的各部分内容亦足以互相支撑，从而形成一个统一的体系。

研究者 可以在本书中学到更多关于如何进行经验研究的知识，并把它们作为自身研究的重要组成部分。此外，我们的目标是：当研究者在进行经验研究时，也可以回到本书，将之作为一份检查单进行逐项检验，同样也可以收获颇丰。

从业者 可以将本书作为一份“食谱”，用于在引进新方法和新技术时对它们进行评估。可以学习如何将经验研究用于日常工作，比如在计划改变开发流程时，可以使用经验研究方法来评估利弊，进而提出建议。

概要

本书主要分为三个部分，其概要总结在表 1 中，该表还给出了本书与最初版本之间的相互对应关系。第一部分中，第 1 章总体介绍了经验研究领域的研究进展，将广义的经验研究和狭义的实验方法均纳入软件工程的研究内容之中；第 2 章总体上讨论了经验策略（包括调查法、案例研究和实验），特别从软件工程的角度阐述了经验研究的内容；第 3 章简要介绍了度量理论和应用；第 4 章概述了如何进行系统文献综述以及如何将几个经验研究中的结果进行综合分析；第 5 章给出了案例研究概述，将其作为相关类型的经验研究；第 6 章则通过引入广义实验过程，重点关注实验部分。

表 1　本书结构

标　题	修订版本	原始版本	主要更新
第一部分　背景			
引言	1	1	
经验策略	2	2	关于重现、综合分析、技术转移和伦理学的新章节
度量	3	3	关于实践中度量的新章节
系统文献综述	4	10[①]	新章节
案例研究	5		新章节
实验过程	6	4	
第二部分　实验过程的步骤			新的运行例子
确定范围	7	5[②]	改编术语
计划	8	6	
操作	9	7	
分析与解释	10	8	
归档与展示	11	9	重大修改
第三部分　实验示例			
实验过程说明	12	11	
视角间真有差异吗	13		新章节
附录			
练习	A	13	将理解型练习移到每一章
统计表	B	A	

① 命名为调查，从不同的视角。

② 命名为定义。

第二部分中，每个实验步骤单独列为一章。其中，第 7 章主要讨论如何确定实验范围；第 8 章介绍实验计划；第 9 章讨论了实验操作；第 10 章给出了分析和解释实验结果的若干方法；第 11 章则讨论了实验的归档和展示。

第三部分包含两个实验例子。其中，第 12 章中的例子用于说明实验过程。第 13 章中的例子则用于说明如何将软件工程实验整理到论文中。

本书将一些练习和数据放在附录 A 中，而将一些统计表放在附录 B 中。这些统计表为本书中的一些例子提供支持。更多更全面的类似统计表在大多数统计类书籍中均可以找到。

练习

本书的练习分为四类。第一类在本书第一部分和第二部分中每章（第 1 ~ 11 章）的结尾，其他三类则在附录 A 中，包括：

理解型练习 每章最后的 5 个问题包含最重要的知识点，其目标是确保读者理解最重要的概念。

训练型练习 这些习题提供了实践实验的机会。特别地，这些习题可以用于分析实验相关数据并回答实验相关问题。

回顾型练习 此练习是针对第 12 ~ 13 章中给出的实验例子的。其目的是回顾已提出的一些实验。在读过文献中的一些实验后，你会发现大多数实验都会有一些问题，主要是因为在软件工程中进行实验时存在着继承问题。我们提出了自己的一些研究例子，而不是简单评论他人的工作。在我们看来，这些研究例子在已发表的实验类型中具有代表性，自然也会有各自的长处和短处。

任务型练习 这些习题用于说明如何使用实验进行评估。这些任务是一些可以在学校或公司的课堂内进行的学习例子，它们只针对一些能够通过简单实验就可以解决的问题。这些任务既可以在读过本书之后完成，也可以在阅读此书时解答。后者提供了一个在阅读本书时可以随时练习的机会。作为替代方案，我们建议老师在自己的专业领域内制定一项任务，此任务可以作为例子用于说明整书每章中提出的概念。

本书基于2000年出版的《软件工程中的实验方法：导论》（Experimentation in Software Engineering：An Introduction）一书。本版是原书的修订扩展版。我们修订了原书中的几个部分，同时也增加了一些新的内容，比如系统文献综述以及案例研究等。

通常而言，一本书的完成并不会仅仅是作者的功劳。对于一本新书而言，来自其他方面（比如家庭成员、朋友、同事、国际同行以及资助机构等）的支持与帮助同样不可或缺。本书也不例外。特别地，我们衷心感谢《软件工程中的实验方法：导论》一书的读者，你们对该书的使用是激发我们出版新版本的动力源泉。同样地，我们也感谢巴西贝南博古联邦大学的 Alan Kelon Oliveira de Moraes 先生给我们发 Email，最终促使了本书的诞生。此外，我们还要感谢以下个人对本书做出的贡献。

首先，我们要感谢本书的第一个主要的外部读者，意大利巴里大学的 Giuseppe Visaggio 教授，他在课堂上使用本书的初稿，并为我们提供了有价值的反馈信息。感谢美国丹佛大学的 Anneliese Andrews 教授以及加拿大渥太华大学的 Khaled El Emam 博士第一时间鼓励我们出版此书并给我们提供了有价值的建议。感谢卢森堡大学的 Lionel Briand 博士，德国曼海姆大学的 Christian Bunse 博士，以及曾在德国凯泽斯劳滕弗劳恩霍夫试验软件工程研究所工作的 John Daly 博士所提供的有关面向对象设计案例的数据。此外，也感谢 Thomas Thelin 博士允许我们将他和本书其中两位作者一起完成的实验囊括进本书中。

本书的前期初稿在 Lund 大学的软件工程研究小组内部进行了使用和评估。我们很感谢该研究小组的成员对本书不同的原始版本给出的反馈意见。特别地，我们想感谢 Lars Bratthall 博士花时间仔细地评阅了初稿并给出了有价值的意见。此外，我们也感谢其他匿名评审人为此书做出的贡献。

对于本书的当前版本，我们收到了许多有价值的直接贡献以及改进建议，鉴于此，我们想感谢美国丹佛大学的 Anneliese Andrews 教授，英国杜伦大学的 David Budgen 教授，英国基尔大学的 Barbara Kitchenham 教授，瑞典布莱金厄技术研究所的 Jürgen Börstler 教授、Samuel Fricker 博士和 Richard Torkar 博士。也感谢 Jesper Runeson 先生将本书转换成 LaTeX 格式。

除了以上人员，我们还要感谢国际软件工程研究网络（International Software Engineering Research Network，ISERN）的所有成员关于经验软件工程研究在一般意义下有意思及有启发性的讨论。

对于案例研究这一章，我们感谢 ISERN 成员以及2007年9月份经验软件工程国际前沿学校的成员对检查单给出的反馈意见。其中要特别感谢 Weyns 博士和 Andreas Jedlitschka 博士对该章前期初稿的评阅。

Lund 大学和布莱金厄技术研究所的许多研究项目多年来为本书提供了持续的贡献。多个基金资助了这些以经验研究为基石的项目，从而帮助我们积累了创作本书的经验。从一定程度上而言，本书是所有这些项目的结晶。

Claes Wohlin 教授
Per Runeson 教授
Martin Höst 教授
Magnus C. Ohlsson 博士
Björn Regnell 教授
Anders Wesslén 博士

推荐阅读

软件工程：实践者的研究方法（第7版）

作者：（美）Roger S. Pressman
译者：郑人杰 等
ISBN：978-7-111-33581-8
定价：79.00元

需求工程：基础、原理和技术

作者：（德）Klaus Pohl
译者：彭鑫 等
ISBN：978-7-111-38231-7
定价：89.00元

软件可靠性方法

作者：（以色列）Doron A. Peled
译者：王林章 等
ISBN：978-7-111-36553-2
定价：45.00元

需求工程：实践者之路（原书第4版）

作者：（德）Christof Ebert
译者：洪浪
ISBN：978-7-111-43986-8
定价：69.00元

软件建模与设计：UML、用例、模式和软件体系结构

作者：（美）Hassan Gomaa
译者：彭鑫 等
ISBN：978-7-111-46759-5
定价：85.00元

推荐阅读

人件（原书第3版）

作者：（美）Tom DeMarco 等 ISBN：978-7-111-47436-4 定价：69.00元

公认对软件行业影响最大、最具价值的著作之一，历时15年全面更新
与《人月神话》共同被誉为软件图书领域最为璀璨的“双子星”，近30年全球畅销不衰

在软件管理领域，很少有著作能够与本书媲美。全书从管理人力资源、创建健康的办公环境、雇用并留用正确的人、高效团队形成、改造企业文化和快乐工作等多个角度阐释了如何思考和管理软件开发的最大问题——人（而不是技术），以得到高效的项目和团队。

设计原本——计算机科学巨匠Frederick P. Brooks的反思（经典珍藏）

作者：（美）Frederick P. Brooks,Jr. ISBN：978-7-111-41626-5 定价：79.00元

图灵奖得主、《人月神话》作者Brooks封笔之作，揭秘软件设计神话！
程序员、项目经理和架构师必读的一本书！

《设计原本》开启了软件工程全新的“后理性时代”，完成了从破到立的圆满循环，具有划时代的重大里程碑意义，是每位从事软件行业的程序员、项目经理和架构师都应该反复研读的经典著作。全书以设计理念为核心，从对设计模型的探讨入手，讨论了有关设计的若干重大问题：设计过程的建立、设计协作的规划、设计范本的固化、设计演化的管控，以及设计师的发现和培养。

第一部分 背 景

第二部分 实验过程的步骤

第三部分 实验示例

附 录

第一部分

Experimentation in Software Engineering

背　景

第1章
Experimentation in Software Engineering

引　言

暂且不论信息技术革命意味着什么，至少它已经让软件成为越来越多产品中的一部分。从烤箱到航天飞机，都可以发现软件的存在。从中可以看出，有大量软件已经被开发出来或者正在开发过程中。软件开发绝不是一件简单的事情；它是一个富有高度创造性的过程。软件领域的迅速发展也使软件项目开发遇到很多问题，比如功能不全、费用超支、逾期完工、质量低下等。这些问题或挑战早在20世纪60年代就已提出，并且在1968年提出了“软件工程”一词，旨在创建一种着眼于软件密集型系统开发的准则。

IEEE［84］对软件工程的正式定义为：*软件工程意味着将系统化的、严格约束的、可量化的方法应用于软件的开发、运行和维护中*。软件工程也在很多著作中被提及和讨论，如Sommerville［163］、Pfleeger和Atlee［134］等。本书的目的是阐明经验研究（Empirical Study）和特定的实验方法非常适用于软件工程领域。上面定义中的三个方面对本书非常重要。首先，它指出软件过程跨越了生命周期的不同阶段；第二，它强调了对系统化的、严格约束的方法的需求；第三，它强调了量化的重要性。经验研究的使用和这三个方面都有关系。1.1节将进一步讨论软件工程背景。1.2节讨论使软件工程更科学化的必要性以及经验研究在此过程中发挥的重要作用。

1.1　软件工程背景

软件过程模型用于描述软件开发时所采取的步骤和执行的活动。软件过程模型有瀑布模型、增量开发模型、演进式开发模型、螺旋模型以及各种敏捷开发模型等，这些模型在一般软件工程书籍中都有介绍。软件过程的示意图如图1-1所示。值得注意的是，无论是开发一个新产品还是维护现有的产品，过程都是至关重要的。

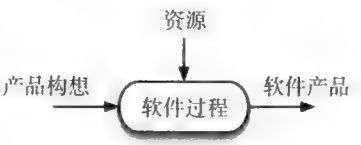


图1-1　软件过程示意图

在图1-1中，产品构想和以人员为主要形式的资源作为软件过程的输入，这些人员在软件过程中通过采取不同的步骤、执行不同的活动来进行软件产品的开发。

软件产品的开发往往是一项非常复杂的工作。由于软件产品的复杂性，软件项目有时可能会经历很长的一段时间（即使使用敏捷开发），并涉及许多人员。这意味着软件过程通常会变得非常复杂，在交付最终产品之前，软件过程包含各种各样的活动，并且需要撰写许多文档。软件过程的复杂性意味着很难对其进行优化，甚至难以找到一个足够好的过程。因此，对企业来说，为了维持其竞争力，一套能够提高其业务水平的方法至关重要。这意味着为了达到改善产品、降低成本等目的，大多数企业都在

不断地努力改进其软件过程。软件过程强调的是需要一种系统规范的工作方式。敏捷方法也不例外。尽管敏捷方法强调不需要太多的文档，强调能够连续地运行代码（而不是仅在一个大项目的后期运行），但是它仍然期望有一种结构化的方法。在改进软件过程时也需要一种系统规范的方法，因此需要一种改进过程的方法。

一个可裁剪并改进软件过程的例子是 Basili [7] 提出的质量改进模式（Quality Improvement Paradigm，QIP），它由若干步骤组成，以支持系统规范的改进方法。在2.9.2节将简单介绍 QIP。一种更通用的改进过程是著名的“计划/执行/研究/行动”环（Plan/Do/Study/Act）[23，42]，下面重点说明一下该改进过程所包含的两种活动（有时会使用不同的术语）：

- 软件过程评估。
- 对软件过程改进建议的评价。

其中，评估用来识别过程改进的机会。现在已有一些用于评估软件过程的模型，其中最著名的模型就是美国卡内基·梅隆大学软件工程研究所的能力成熟度模型 CMM [33，130]。该评估模型有助于准确定位需要改进的地方。CMM 定义了五个成熟度级别，每个级别上都有对应的关键过程域。一般建议企业根据其成熟度级别来重点考虑对应的改进部分。

假如可以通过某种形式的评估来识别需要改进的过程域，那么下一步就要确定怎样定位这些改进域来解决已经发现的问题。比如，如果在系统测试时发现太多缺陷，那么可能就要改进早期的测试、审查甚至某个开发阶段（比如，软件设计）。其目标是通过对当前现状的评估以及对当今先进水平的了解，给出过程改进的具体建议。在确定改进建议后，就需要确定引入哪些改进。通常，不能仅改变当前的软件过程，而不先评价改进的实际效果。换而言之，在做任何大的改变之前都需要对这些改进建议进行评估。

随之产生一个问题，即如果没有人员的直接参与，对过程改进建议的评估是很难进行的。对于产品来说，或许可以事先建立一个原型来评估该产品运行时的情况。但对于过程而言，则难以构建这样的原型。通常可以通过仿真和比较的方法来对过程进行评估，但这仍然是基于模型的评估。真正对过程或者过程改进建议进行评估的唯一方法就是让人们去使用它，否则在人们使用它之前它就仅仅是一个描述。经验研究是对过程和基于人的活动进行评价的一种主要方法，同时也是用于评价软件产品或工具使用情况的常用方法。实验为评估基于人的活动提供了一种系统化的、规范的、可量化和可控制的方法。这也是经验研究通常用于社会及行为科学的主要原因，参见 Robson 的例子 [144]。

同时，经验研究尤其是实验的方法，对于软件工程领域的研究人员而言也是相当重要的。新的方法、技术、语言和工具都不应该仅仅只是提出来、发表、投入市场那么简单。将这些新的发明和建议与现存的方法进行比较，来对其进行评估也变得尤为重要。实验的方法提供了这种机会，因此应该利用起来。换而言之，在进行软件工程

研究时，我们应该学会使用这些已有的方法和策略。下面将会详细介绍。

1.2　科学与软件工程

软件工程是一个跨学科的学科。它从技术问题（如数据库和操作系统）跨越到语言问题（如语法和语义），再到社会问题和心理学。软件开发是人力密集型的活动，
5 至少时至今日我们尚不能“机械化地制造”出新的软件。它是一个基于在该领域工作的人的创造力和聪明才智的学科。即便如此，在学习和研究软件工程时，我们仍然将其看作一门科学学科。这意味着针对软件开发方法的改变，我们应采用科学的方法开展研究和做出决定。

为了在软件工程领域进行科学研究，我们必须理解这些已有的方法、它们的局限性和适用条件。软件工程源于技术范畴，因此自然会考虑到研究中已使用的方法，如硬件设计和编码理论，但是基于软件工程跨学科的特性，我们还应该考虑其他学科的研究方法。Glass 总结了软件工程领域的四种研究方法［62］，而将这些方法用于软件工程领域则是 Basili 最早提出的［9］。这些方法如下。

- 科学方法（Scientific）：首先观察世界，并在观察的基础上建立模型，如仿真模型。
- 工程方法（Engineering）：首先研究当前解决方案，然后提出修改建议，并进行评估。
- 经验方法（Empirical）：首先提出一个模型，然后通过经验研究对其进行评价，例如，案例研究和实验的方法。
- 分析方法（Analytical）：提出一个正式的理论，然后与经验观察的结果进行比较。

工程方法和经验方法可以被视为科学方法的变体［9］。

传统上，分析方法常用于电气工程和计算机科学中更形式化的领域，例如电磁理论和算法。科学方法也应用于应用领域，如仿真一个电信网络以评价其性能。然而，需要指出的是，仿真不仅仅是科学方法中的一种手段，仿真同样可以作为实验的一种手段。工程方法主要应用于工业界。

经验研究历来被用于社会科学和心理学研究，在这些领域我们无法像在物理学中那样阐述自然规律。[⊖]社会科学和心理学关注的是人的行为。因此，在这样的背景下得出的重要结论是：软件工程是与开发软件的人的行为密切相关的。因此，除了在特定的技术领域中，我们不能指望在软件工程中发现任何形式化规则或规律。本书的重点是将经验研究应用于软件工程，其目的是在实施通用的经验研究和特定实验时强调其
6 基本过程。本书提出了一个实验过程，强调了实施实验的各个基本步骤，给出了实施指南，并通过软件工程领域的案例展示各个步骤。

⊖ Lehman［110］提到了软件演化规律，但这个概念一直没有在后续理论工作中得到广泛应用，见 2.7 节。

必须指出的是，本书不是说分析方法、科学法和工程化方法不适合软件工程领域。它们对于软件工程同样是必需的，例如，我们可以为软件可靠性增长建立数学模型[116]。此外，这些研究方法并不是正交的，比如，可以在工程方法中适当采用经验研究的方法。重要的一点是，我们应该在经验研究中更好地利用其他已有方法，这些方法常出现在其他学科中，如行为科学等。软件工程的本质使得它与工程领域之外其他的非技术学科有很多共同之处。

根据 Zendler［182］的说法，软件工程中的第一个实验是由 Grant 和 Sackmann［69］在 20 世纪 60 年代末进行的在线和离线测试实验。在 20 世纪 70 年代，一些先驱针对结构化程序设计［115］、流程图［151］和软件测试［126］进行了实验。Basili 等［15］在 80 年代中期强调了系统化的实验方法在软件工程中的重要性。另一些文章则强调了在软件工程中经验方法的重要性，参见 Basili、Fenton、Glass、Kitchenham、Pfleeger、Pickard、Potts 和 Tichy 等的著作［9，57，62，97，140，169］。之后，Tichy 等［170］、Zelowitz 和 Wallace［181］、Glass 等［63］指出在软件工程研究中缺乏实验性证据。后者在其出版物中指出，在软件工程研究中仍然存在过多的宣传式研究[140]。因此软件工程需要更科学的研究方法。本书的重点是软件工程和经验研究在软件工程中的应用，尤其是实验方法在软件工程中的应用。Sjøberg 等人的调查［161］表明，在软件工程领域已发表的实验数目在逐步增加，现在已发表了大量的实验。

软件工程中的经验策略包括：

- 建立正式的实验；
- 研究工业界的真实项目，如实施案例研究；
- 进行调查研究，比如访谈。

这些策略将在本书的第 2 章和第 5 章详细介绍，之后本书将重点讨论实验的方法。关于这些研究策略的更一般性介绍可参见 Robson［144］；关于广义案例研究的详细介绍可参见 Yin［180］；针对软件工程领域的案例研究方法可进一步参考 Runeson 等人的文献［146］。这些研究策略既不是完全正交，也不是竞争关系。只是给出了一个简单的分类，有些研究方法可以被看作是这些策略的组合，或者介于某两种方法之间。因此，不同策略之间也存在相同之处，不是严格意义上的分类。

在软件工程中运用实验方法的主要原因是希望了解和识别不同因素或变量之间的关系。大量先入为主的观念存在，但它们是真的吗？面向对象的方法能够提高重用性吗？代码审查的性价比如何？我们是否需要开评审会议，或者是否将评论交给主持人就够了？这类问题都需要进行调查，以提高我们对软件工程的理解。提高认识是不断改变和改进我们工作方式的基础，因此通用的经验研究和特定的实验方法都非常重要。

7

对该领域的介绍以对一个实验过程的介绍为基础，该过程中的基本步骤也适用于其他类型的经验研究。然而，关注焦点是为实施软件工程实验提供指导和支持。此外，这里需要说明的是“真正的”实验，即完全随机的实验，这在软件工程中难以实现。软件工程实验往往是准实验，即，实验中不可能随机分配实验的参与者［37］。准实

验可以提供有价值的结果，因而也是非常重要的。本书讨论的过程包括“真正的”的实验和准实验。后者将在介绍实验威胁时全面讨论。

本书的目的是介绍经验研究和实验方法，进一步强调在软件工程领域中做实验的机会和好处。经验研究方法能够也应该在软件工程领域中得到更多的应用。Tichy 等人[169]反驳了那些反对在软件工程中应用经验研究方法的言论。希望本书能够有助于在软件工程研究和实践领域采用经验研究和实验的方法。

1.3　练习

1.1　为什么实验可被视为过程改变的原型方法？

1.2　如何利用实验进行活动改进？

1.3　为什么经验研究在软件工程中很重要？

1.4　经验研究方法在什么情况下最适用于软件工程？请分别与科学方法、工程方法和分析方法进行比较。

1.5　经验方法分为哪三个策略？

第2章
Experimentation in Software Engineering

经验策略

两种不同的研究范式（探索性研究和解释性研究）通常采用不同的经验研究方法。*探索性研究*着重研究自然环境中的对象，从观察中发现结果。这意味着需要一种*柔性研究设计*［1］来应对观察现象的变化。柔性设计（Flexible Design）研究也称为*定性研究*（Qualitative Research），因为该研究主要从定性数据中获取信息。归纳研究试图利用人们已经提出的理论来解释现象。它的关注点在于发现研究中主体观察到的起因，并理解他们当前看待问题的观点。这里，主体指为了评估对象而参与经验研究的人。

*解释性研究*主要关注量化关系或者把两个或多个分组和目标进行比较以确定因果关系。该研究往往通过建立受控实验来进行。这类研究是一种*刚性设计*（Fixed Design）［1］研究，即在研究开始之前各因素就已确定。刚性设计研究也称为*定量研究*（Quantitative Research），因为该研究主要从定量数据中获取信息。比如，测试一些操作或者活动的效果时可以采用定量调研。定量数据的优点是可以进行比较和统计分析。定性和定量研究可以用在同一个研究课题中以处理该课题中不同类型的问题。比如，定量研究可以用于分析一个新的审查方法是否减少了测试中发现错误的个数，而定性研究则可用于回答不同审查组之间差异的来源。

如前所述，刚性设计策略（如受控实验）适用于测试某种处理方法的效果，而信念、理解和各种观点展示的柔性设计研究适用于分析定量研究结果产生的原因。这两种研究方法是相互补充而不是相互排斥的。

本章的目标：①介绍经验研究策略；②强调一些与经验策略有关的重要方面；③举例说明如何在技术转让和改进中使用策略。为了达到第一个目标，2.1 节对经验策略进行了概述，并详细讨论了调查、案例研究和实验。2.2 节～2.4 节简要介绍了几种不同的经验策略。2.5 节对这些策略进行了比较。2.6 节通过采用重现实验的方法来达到第二个目标。2.7 节简要地讨论了和经验研究有关的理论，2.8 节介绍了经验研究的证据汇聚。最后，2.9 节阐述研究策略在技术转让过程中的使用，并成为程序改善的一部分。

2.1 经验策略概述

根据经验研究的条件和评估目的（针对技术、方法或者工具）的不同，有三类主要策略：*调查法*、*案例研究*和*实验*［144］。

定义 2.1 调查法（Survey）是一种通过收集来自于人或者与人有关的信息，来描述、比较或者解释人们的知识、态度和行为的方法［58］。

调查法常常是一种回顾式的调查过程，因此可以在一种工具或者技术已经使用一段时间之后采用此方法［133］。其收集定性或定量数据的主要手段是访谈或者调查问卷。数据采集往往来自于待研总体的代表性样本，通过分析调查结果可以得到描述型和解释型的结论，然后再将这些结论推广到样本所属总体。关于调查法的深入讨论参见 Fink［58］和 Robson［144］。

定义 2.2　软件工程中的**案例研究**（Case Study）是一种经验性探究方法，它通过多个证据源来调查在真实环境下当前软件工程现象的一个实例（或少量的实例），尤其用在当现象和环境的边界难以清晰界定的时候［146］。

案例研究可用于研究项目、活动或者任务。在研究过程中有目的地收集有关数据，然后采用统计分析的方法对所收集的数据进行处理。案例研究常常用于跟踪某个特定的属性或者建立不同属性之间的联系。其控制程度低于实验方法，因为案例研究属于观察性研究，而实验是一种受控研究［181］。例如，案例研究可用于构建一个模型以
10 预测测试中错误的数量［2］。这类研究常常会用到多元统计分析，包括线性回归和主成分分析［118］。针对广义案例研究的深入讨论参见 Robson［144］、Stake［165］和 Yin［180］等，针对软件工程的案例研究详见 Pfleeger［133］、Kitchenham 等［97］、Verner 等［173］、Runeson 和 Höst［145］、Runeson 等［146］。

对于本书中重点讨论主要的经验研究策略——实验，我们的定义如下。

定义 2.3　软件工程中的**实验**（Experiment，或称受控实验）是一种经验性探究方法，它操控研究情境中的一个因素或者变量。在保持其他变量不变的前提下，通过随机选择的主体实施不同的方案，或者由不同的主体来实施同样的方案，然后测量对输出变量的影响。在面向人的实验中，人们在对象上使用不同的方案；而在面向技术的实验中，则在不同的对象上使用不同的技术方案。

实验大都是在实验室情境下进行的，以保障高度的可控制性。实验中，主体被随机分配给不同的方案。目标是操控一个或多个变量，并保持其他变量相对稳定。然后测量该操控的结果，并进行统计分析。当不能随机地将方案分配给主体时，我们可以采用准实验的方法。

定义 2.4　**准实验**（Quasi-Experiment）是一种类似于实验的经验性探究方法。在准实验中，方案到主体的分配不是随机的，而是根据主体或者对象本身的特征来选择的。

在实验研究中，统计推断的方法是通过统计显著性来说明一个方法比另一个方法好［125，144，157］。统计方法将在第 10 章中讨论。

调查法在社会科学中很常见，如通过民意调查来判断下一届选举中公众会如何投票。尽管调查法也可以对相似情况进行比较，但是它不会提供对执行或者测量的控制，更不可能像其他策略一样操控变量［6］。

案例研究可以找出或许对结果有影响的关键因素，并且整个活动会被记录下来

[165, 180]。案例研究是一种观察性方法，也就是说，它是通过观察项目或者活动的进展情况来完成的。

实验是一种正式、严谨且受控的调查。在实验中，关键因素被识别且被操控，而实验情境中的其他因素保持不变，详见6.1节。案例研究和实验的区别在于对环境的控制程度[132]。采用实验研究时会刻意在不同场景下执行操作，以找出两种场景下的区别，如一个受控场景和调查中的实际场景。被操控的因素可以是审查方法或者软件开发者的经验等。在案例研究中，调查情境被研究中的实际项目所控制。 11

根据对调查研究的设计，研究策略可以获得定性或者定量的数据，如表2-1所示。调查法的类型取决于调查问卷的设计，即收集什么数据以及是否可以采用统计分析方法进行数据分析都由设计决定。案例研究也是如此，不同点在于调查法属于事后回顾，而案例研究则在项目执行过程中进行。当然调查法也可以在项目执行前启动，尽管此时调查的目的是为即将开展的项目提供一些想法，但它仍然是基于先前的经验，并通过回溯先前经验来完成的。

表2-1 经验策略中的设计类型和定性/定量数据

策 略	设计类型	定性/定量
调查法	刚性	定性/定量
案例研究	柔性	定性/定量
实验	刚性	定量

实验基本上是纯定量的，因为实验主要关注对各种变量的度量。改变这些变量，然后再度量。在实验期间，收集定量数据并采用统计方法对其进行分析。当然，也可以收集定性数据来对数据进行解释[93]。以下各节将对每种经验策略分别进行介绍。

2.2 调查法

对于已经使用的技术或工具[133]，或者在引入它们之前，常采用调查法。可将调查法看作抓取现状的快照，例如，调查法可用于民意测验和市场研究。

在软件工程领域，例如可以采用调查法研究一个新的开发过程如何通过提高开发人员的质量意识来实现质量保证或排序质量属性[94]。可以从公司的所有开发人员中选取一些开发人员作为样本，让他们来回答调查问卷以获得研究所需的信息。最后将收集到的信息整理成可定量或定性处理的形式。

2.2.1 调查法的特征

抽样调查的目的不是为了理解特定的样本，而是为了理解被抽样的总体[6]。例如，通过访谈25名开发人员对新过程的看法，可以估计公司中100名开发人员的看法。调查法的目的是得到普遍性结论。 12

调查法可以评估大量变量，但有必要努力通过最少的变量获得对总体的最多了解，

因为减少变量也就简化了数据收集和分析工作。调查法中过多的问题会使应答者烦于回答，从而导致数据质量下降；而另一方面，调查法的目的是得到一个宽泛的概貌，这又不得不涉及多个领域的问题。

2.2.2　调查法的目的

调查法的一般目标可以分为以下三种［6］：

- 描述型（Descriptive）；
- 解释型（Explanatory）；
- 探索型（Explorative）。

描述型调查法的目的是给出一个关于总体的断言。该方法可以确定某些特性或属性的分布。这里，我们不关心为什么会存在观测到的分布，而关心到底是什么分布。

解释型调查法旨在解释一个总体。例如，当研究开发人员如何使用某种审查技术时，可能需要解释为什么一些开发人员喜欢这种技术，而其他人更喜欢另一种技术。通过考察不同备选技术之间的关系和一些解释性变量，或许可以尝试着解释为什么开发人员会选择某一种技术了。

最后，探索型调查法常用作详尽研究之前的预研究，以确保将来不会忽略任何重要问题。一般来说准备一个松散结构问卷，并从总体中抽样调查就可以了。收集和分析调查所得的信息，其结果用于之后的详尽研究。换言之，探索型调查不解决基础研究问题，但它可以提出一些值得分析的新问题，以便在后续的特定研究或深入研究中考虑。

2.2.3　数据收集

最常见的两种数据收集方式是问卷调查和访谈［58］。问卷可以是纸质的，也可以是某种电子形式，如电子邮件或网页。通过问卷调查收集数据的基本方法是将调查问卷连同问卷填写说明一起分发出去，应答者填完问卷后将其反馈给研究者。

13

让访谈者处理问卷调查（通过电话访谈或面对面访谈）而不是由应答者自己填写，这种做法有如下优势：

- 访谈调查会得到比邮件调查等更高的回复率。
- 访谈通常会降低“不知道”和“无应答”的数量，因为访谈者可以回答问卷中的问题。
- 访谈者不仅可以观察，还可以提问。缺点是耗财、耗时，当然这取决于样本的数量和调查的目的。

2.3　案例研究

案例研究是指在特定的时间限制内，对真实环境中的单个实体或现象进行研究。多数情况下，现象可能难于从其所处环境中被明确区分。例如，研究人员在一段持续

的时间内收集单个项目的详细信息。在进行案例研究时，需要采用多种数据收集方法和不同分析角度［146］。这里简要介绍不同类型经验策略的适用情况，更详细的介绍将在第5章展开。

举例来说，如果想要比较两种方法，需要根据评估的尺度、分离各因素的能力以及随机化的可行性来决定是使用案例研究还是实验方法。案例研究方法的一个例子是，通过使用试点项目，将试点项目的结果和某些基线进行比较来评估试点的效果［97］。

案例研究非常适合对软件工程方法和工具在工业界的使用情况进行评估，因为它可以避免按比例放大问题。案例研究和实验的不同之处在于，实验样本所包含的变量是人为控制的，而案例研究则是在可以代表典型情况的变量中进行选择。案例研究的优势在于它更容易设计并且更加真实，劣势则体现在它的结果难以归纳且更难解释。也就是说，要说明一个典型情况下的某个结果比较容易，但想要把它推广到其他情况还需要更多研究［180］。

当某个过程的变化会造成广泛影响时，案例研究是一种合适的研究方式。变化所带来的影响只能在高度抽象的层次上进行评估，因为在整个开发过程中，这个变化会包括很多细小而具体的变化［97］，并且该变化带来的影响无法立即明确。例如，如果我们想要了解一种新的设计工具是否能够提升可靠性，有可能必须要等开发的产品发布后才能评估它对运行失效的影响。

14

案例研究是很多学科进行经验研究的一种常规方法，如社会学、医学和心理学。在软件工程领域，案例研究不仅可以用于评估某个现象如何发生或者为什么发生，还可以用来评估差异性，比如两种设计方法间的差异。换言之就是评估在特定情况下哪一种方法更加合适［180］。例如，调查使用基于视角的阅读方法是否能够提高需求规格说明的质量就是软件工程领域中的一个案例研究。像这样的研究并不能像测试一样确定基于视角的阅读减少了多少错误，因为还需要一个不使用基于视角技术的对照组，但是它能帮助我们理解该机制在审查中所起的作用。

2.3.1 案例研究的安排

案例研究可以作为一种比较研究的策略，用来比较采用某种方法或者操作形式所产生的结果和采用其他方法所产生的结果。为了避免偏差并确保内部有效性，必须为评估案例研究的结果建立一个稳定的基准。为此，Kitchenham 等人提出了三种安排案例的方法［97］：

- 一种解决办法是把使用新方法所产生的结果与公司基线做比较。公司应从其常规项目中收集数据，并计算出诸如平均生产率和缺陷率等特征数据。这样才可能将案例研究的结果和公司的基线数据进行比较。
- 可以选择姊妹项目作为基线。被研究的项目采用新方法，而其姊妹项目使用已有方法。两个项目应具有相同的特征，也就是说，两个项目必须是可比较的。

- 如果某个方法要在个别产品组件上应用，那么它也应该可以被随机地运用到某些相关组件上。这一点与实验非常相似，但由于这个项目并不是从总体中随机抽取的，所以它并不是实验。

2.3.2　混杂因子和其他方面

在进行案例研究时，必须设法最小化混杂因子所带来的影响。混杂因子（Confounding Factor）是指无法将两个因素带来的影响相互区分开的一个因子。这一点非常重要，因为在案例研究中，我们并没有实验中的控制手段。举例来说，判别一个较好结果的产生是依赖于工具还是工具使用者的经验，可能会十分困难。混杂效应（Confounding Effect）可能会在我们学习如何使用某个方法或工具以评估其带来的好处时带来问题，或是在决定是选择有激情的员工还是有怀疑精神的员工时带来困扰。

15

案例研究本身也是有利有弊的。案例研究的价值在于它包含很多实验无法体现的特质，比如，规模、复杂度、不可预测性以及动态性。而案例研究也存在着以下潜在问题。

- 一个小型或者简单的案例研究并不适合发掘软件工程原理和技术。规模的增长将会导致最具参考价值的问题类型发生变化。换言之，尽管我们的目的是研究同一个问题，但这一问题在小型案例研究和大型案例研究中可能变得不同。比如，在小型案例研究中，主要问题可能就是要研究的技术，而到了大型案例研究中，主要问题可能就变成了参与人数以及随之而来的人员沟通问题。
- 研究人员并不完全控制案例研究情况。从某个角度来看，这也是一件好事，因为无法预测的变化常常可以让研究人员更了解被研究的问题。真正的问题在于我们无法确定混杂因子带来的影响。

第5章对案例研究有更详尽的阐述。

2.4　实验

当希望控制某种情境并直接、精确和系统性地操控其行为时，实验是一种较好的选择。实验往往采用多种处置，并比较其结果。例如，如果可以控制谁在使用这个方法，谁在使用其他方法，同时还能控制使用方法的时间和地点的话，就可以完成一个实验了。这种类型的操作可用于离线（off-line）场景，如在可控的实验室里模拟真实世界中发生的事件。实验也可以选择性地用于在线（on-line）场景，这意味着调研将在真实环境中执行[6]。在线环境中实施控制更为困难，虽然很多因素不可控，但仍然有一些因素是可控的。

实验可以是*面向人的*（Human-Oriented）和*面向技术的*（Technology-Oriented）。在面向人的实验中，人们在对象上实施不同的处置，例如对两段代码采用不同的代码审查方法。在面向技术的实验中，通常将不同的工具用于相同的对象，例如对于相同的程序使用两个不同的测试用例生成工具。面向人的实验比面向技术的实验更不易控制，

因为人们在不同的场合会有不同的行为，而工具（大部分）是确定的。再者，考虑到学习效果，一个人不可能将两种方法等效地用到同一段代码中，而两个工具则不存在这种偏差问题。 16

正如前面所提及的，考虑上下文环境更易于清楚地说明案例研究和实验方法之间的区别。所谓不同的上下文环境[⊖]可以是不同的应用领域和系统类型［132］。在实验中，需要确定感兴趣的上下文环境、其中的变量和覆盖这些内容的样本。这意味着我们需要选择能代表实验机构多种典型特征的对象，之后设计实验来测量每个特征的值（一个以上）。例如，通过对不同语言编写（也可能是从一个编程语言转移到另一种编程语言）的两个不同系统进行测试所发现的错误，来调查一个审查方法的效果。这时，不同的系统就是评估该审查方法的上下文环境，当然实验中还需要相似的对象；而审查方法就是独立变量，不同程序设计语言编写的程序是被研究对象。

实验的设计应使得所涉及的对象能代表我们感兴趣的所有方法。并且，尽可能考虑将当前状况作为基线（控制），也即基线代表了独立变量的一个水平（或者值），而新情形则是想要评估的另一水平。这时，新情形下独立变量的水平描述了被评估情形与控制基线的差异。当然其他所有变量的取值应保持不变，如应用领域和编程环境。

2.4.1 特征

实验方法适用于调查不同的内容［72，162］，包括：

- 确认理论，即检验已存在的理论。
- 确认传统观点，即检验人们的观念。
- 探索关系，即检测是否存在某种关系。
- 评估模型的准确性，即检测模型的准确性是否符合预期。
- 验证度量方法，即确保一个度量方法确实度量了它应度量的东西。

实验的优势在于，它可以调查在哪些情形下论断是正确的，也可以提供一个上下文环境，推荐使用其中的某些标准、方法和工具。 17

2.4.2 实验过程

实施一个实验涉及几个不同的步骤。这些步骤如下。

（1）确定范围；

（2）制定计划；

（3）操作；

（4）分析和解释；

（5）归档与展示。

这些实验过程将在第6章介绍，而其中的每一个步骤将在第7～11章中详细讨论。

⊖ 也译为“情境”。——译者注

2.5　经验策略比较

调研的前提条件限制了研究策略的选择。对策略的比较可以基于很多不同的因素进行。Pfleeger 提出了一些比较因素［133］，本书补充后如表 2-2 所示。下面将逐一阐述。

表 2-2　研究策略因素

因　素	调　查	案例研究	实　验
执行控制	无	无	有
度量控制	无	有	有
调研成本	低	中	高
重现难易度	易	难	易

执行控制（Execution Control）描述了研究者对整个研究的控制程度。例如，在案例研究中，数据收集必须在项目执行中进行。如果项目经理因为经济等原因决定终止被研究项目，研究者则无法继续完成该案例研究。而实验研究则相反，研究者控制实验的执行。

度量控制（Measurement Control）是指研究过程中，研究者能在多大程度上决定可以收集哪些度量内容、包括或者排除哪些度量。例如如何收集需求易变性数据。一些在调查法中无法做到的度量控制，在案例研究或者实验中却是可能的。因为在调查法中，只能搜集到相关人员对需求易变性的看法。

与上述因素紧密相关的是调研成本。选择不同的策略，成本是不一样的。这和研究的规模、所需的资源等因素有关。成本最低的策略就是调查法，因为它并不需要大量的资源。案例研究和实验的区别在于，案例研究中，如果研究的对象是一个项目，则该项目的输出可能是某种可以投入销售的产品，即这是一种在线调研。而在离线实验中，其输出是某种形式的经验或知识，不可能像产品那样可以直接盈利。

另一个需要考虑的重要方面就是重现该研究的可能性。重现的目的是为了验证原研究中的结论在更大的总体中仍有效。当设计和结果都能被重现时，该重现就称为“真”重现。通常来说，重现调研的结论不会与原调研结果差别太大。

18

从研究的时间维度看，与重现有关的是纵向研究（Longitudinal Study）［141］。两者之间的主要差异在于，纵向研究主要是采用同样的主体，而重现研究多数都是针对新主体。换句话说，重现意味着多个调研，而纵向研究则是一个调研。纵向研究往往会持续一段时间。例如，一个调查法可以在多个场合进行；实验能够重复进行；案例研究如果持续一段时间，也可以看成是纵向的。纵向研究通常是为了理解、描述或者评估某些随时间变化的事情［144］。

经验策略的选择取决于调研的前提条件、调研的目的、可用的资源和采用的数据分析方法。Easterbrook 等人在［50］中给出了更多的策略选择建议。其实，不同研究类型之间的界线并不总是那么清晰。例如，比较型案例研究也可以看作是工业环境下

的准实验；软件工程课程结果的事后观察性研究也可以看作是学生实验。

2.6 重现

实验的重现（Replication）是指在相似条件下重复实验过程，例如只改变实验主体。该方法有助于研究人员确定实验结果的置信度。如果随机性假设是正确的，也就是说实验主体是总体的典型性代表，那么在该总体内重复实验应该得到与之前实验相同的结果。如果无法得到相同的结果，说明在实验设计中没能考虑所有会影响结果的因素。即便能度量某个变量或重复实验，也可能很困难或成本很高。 19

重现有多种类型［89，155］：

- *准确重现*（Close Replication）：尽可能准确地遵循原实验规程，该类型有时也称为*精确重现*（Exact Replication）［155］。
- *差异重现*（Differentiated Replication）：指使用不同的实验规程来研究相同的研究问题，在实验中研究人员会有意地改变一个或多个主要条件。

Basili 等人［20］提出了一个更细粒度的分类：

- 严格重现（Strict Replications）（与准确和精确重现同义）；
- 变更研究中变量的实验重现；
- 变更研究关注点中变量的实验重现；
- 变更实验环境中的上下文变量的实验重现；
- 变更实验运行方式的实验重现；
- 扩展理论的实验重现。

其他研究领域也使用了许多不同的分类标准［64］，且这些研究领域间不存在标准化术语。同样，在软件工程领域也没形成标准术语。上文中给出*准确*和*差异*重现划分是对软件工程领域中实验重现的首次分类。

准确重现的优点是已知因素保持受控状态，从而可以保证输出结果的置信度。然而，有些准确重现需要相同的研究人员来进行，因为实验规程中的一些隐性知识很难被记录并文档化［153，154］。另一方面，准确重现中实验者的主观偏见也是一个潜在风险［95］。此外，有人质疑在软件工程研究领域是否存在真正的准确重现，因为在软件工程复杂的实验配置中，很多因素都可能发生变化［89］。

另一方面，差异重现更多地在探索型研究中使用。如果能很好地记录和分析各因素和实现设置的差异，那么就可以通过重复研究获取更多的知识。在差异重现中要考虑和记录的因素包括：

- 执行实验的地点；
- 执行实验的实验者；
- 实验的设计方案；
- 实验工具，即表或其他材料；
- 被度量的变量；

- 执行实验的主体。

第 8 章将详细讨论这些因素。争议的出现是针对重现最初的假设，而不是针对特
20 定的实验设计［123］，也就是说相对于准确重现，更支持差异重现。

2.7　软件工程理论

“理论给出对基本概念和基本原理的解释与理解，它是关于事物发展趋势和现象的知识。”［72］如前所述，实验可以形成、确认和拓展理论。然而，Hannay 等人在回顾 1993～2002 年软件工程实验的系统文献综述［72］中指出，在软件工程中，极少使用理论。该综述共考察了 113 篇文献，在其中的 23 篇中发现了 40 条理论，其中仅有两条理论被一篇以上的文章使用。

Endres 和 Rombach［53］提出了软件工程中的 50 条“定律”。所谓定律是指用来描述自然科学环境中可重复现象的概念，他们将这一概念应用到软件工程中。其中给出的许多“定律”已超越了软件工程范畴，如，“从初学者成为专家需要 5000 个小时”。他们认为，“理论”解释“定律”，“假设”是对观察到的现象给出的一个试探性解释，而“猜想”则是对现象的一种猜测。Endres 和 Rombach 列出了出现在软件工程文献中的 25 个假设和 12 个猜想。

Zendler［182］采用另一种方法，定义了一个“初步的软件工程理论”，由 3 个根本假设、6 个核心假设和 4 个基本假设组成。这些假设是分层次的，根本假设最抽象，基本假设最具体，直接来源于实验研究的成果。

Gregor［70］介绍了 5 类通用理论，这些理论可能适用于软件工程领域［72］：

（1）分析：这类理论描述研究对象，如分类学、分类法以及本体论等。

（2）解释：这类理论着重于解释事物，比如，为什么发生了某件事。

（3）预测：这类理论旨在预测将会发生什么，例如使用数学或概率模型。

（4）解释及预测：这类理论结合第 2 类和第 3 类理论，通常被称为“基于经验的理论”。

（5）设计和行动：这类理论描述如何做事，通常应用于设计学科［76］。但是否属于一类理论尚存在争议。

Sjøberg 等人［162］提出了一个软件工程理论框架，由以下四个主要部分组成：

- 结构；
- 命题；
- 解释；
- 作用域。

21

结构是描述理论的实体，理论根据前面的分类对其进行描述、解释或者预测，命题由结构之间的关系组成。解释是对命题（即结构之间的关系）的逻辑推理或经验观察。

理论的作用域定义了理论可用的范围。Sjøberg 等人［162］提出了四个原型类来描述作用域，即参与者、技术、活动和软件系统，如表 2-3 所示。

表 2-3 Sjøberg 等提出的软件工程理论框架

原 型 类	子 类
参与者	个人、团队、项目、组织或者行业
技术	过程模型、方法、技术、工具或语言
活动	设计、创建、修改或分析（一个软件系统）
软件系统	软件系统可以从多个维度进行分类，如：大小、复杂性、应用领域、商务/科研/学生项目，或者行政管理/嵌入式/实时系统，等等

尽管从理论观点来考虑是有吸引力的，但是这些理论系统在软件工程领域至今还未形成任何影响力。理论对于研究领域中知识的概念化以及交流是很重要的，对于聚合已有研究和建立重现研究也很有用。无论是技术的策略选择，还是基于预测模型的项目决策，做决策时都可以用理论来和从业者进行交流。因此，为了尽早成为一个成熟的科学领域，应该鼓励软件工程中的理论构建。

2.8 经验研究的证据汇聚

随着经验研究数量的增长，需要汇聚多个经验研究取得的证据，比如，重现研究。首先，研究应该建立在彼此的基础上，新的研究应该考虑将现有知识作为起点。其次，有些问题需要多个经验研究共同给出答案，任何独立的研究都不足以单独回答这些问题。经验证据的收集和综合本身也必须符合科学标准。

系统文献综述（Systematic Literature Review）是收集和综合不同来源的经验证据的重要手段。Kitchenham 和 Charter 将系统文献综述定义为："它采用一套良定义的方法， 22
以一种无偏见的、（一定程度上的）可重复的方式去识别、分析和解释所有与被研究问题相关的证据的一种二次研究形式" [96]。被检索到的经验研究称为*原始研究*（Primary study），系统文献综述被称为*二次研究*（secondary study）。Kitchenham 和 Charter 为该类综述的开展给出了指南，详见第 4 章。

和其他经验研究类似，每个系统文献综述也有一个特定的研究问题。其研究问题与所综述的经验研究的结果相关，通常都是这种形式："A 技术/方法是否比 B 更好（或更差）"[106]。

经验研究的搜索通常采用基于关键字的数据库查询、期刊/会议文集/灰色文献（如技术报告）检索等方式 [96]。"滚雪球（Snowballing）"过程是指通过一篇论文的参考文献去找到其他的相关论文，反之亦然 [145]。注意滚雪球既可以是后向的也可以是前向的。后向就是查询该论文的参考文献，而前向就是寻找哪些相关的论文引用了该篇论文。

如果研究问题是很通用的问题，或者对于该领域的研究成果较少，可以采用*映射研究*（Mapping Study）的方法（也称为*概览研究*，Scoping study）。映射研究针对更宽泛的研究问题，目的是识别某个主题的实践或研究现状，尤其是识别研究趋势 [106]。由于涉及范围较广，搜索和分类规程没那么严格，有更多定性的特征。

系统文献综述和映射研究对于检索到的研究是否应该纳入其中都必须有明确的判别准则和分类标准。对于系统文献综述而言，一个自然的标准就是研究应该是经验研究，而映射研究也可以包括非经验研究。

当针对一个主题的经验研究被收集起来后，就可以进行综合或聚合了。基于统计方法的综合称为元分析（Meta-analysis）。软件工程中元分析的例子有缺陷检测方法［74，121］、敏捷方法［46］和结对编程［73］。

如果不能用元分析法，则可以使用描述性综合法，包括数据的可视化、数据表格和数据的描述性统计［96］。文献综述研究的问题越宽泛，其证据综合就需要越多的定性方法。Cruzes 和 Dybå 发表过一篇有关定性综合方法的综述论文［39］。

软件工程领域的系统文献综述在 21 世纪的头 10 年有了实质性增长。Kitchenham 等称在 2004～2008 年期间发表了 53 篇不同的系统文献综述［103，104］。这些综述除了进行经验发现的综合之外，也对经验研究的报告方法以及存储经验研究的数据库给出了改进建议。

23　第 4 章将对系统文献综述进行更深入的阐述。

2.9　软件工程领域的经验主义

为什么要在软件工程领域开展实验和其他经验研究呢？主要原因是采用定量的经验研究方法可以为软件开发的理解、控制、预测和改善提供客观且具有统计学意义的结果。对于寻求改进的组织而言，经验研究是其做决策时的重要依据。

在引入一种新技术、新方法或其他工作方式之前，最好针对这个改进的优点做一次经验评估。本节将介绍一个评估软件过程变化的框架，针对桌面、实验室和开发项目三种情境推荐不同的经验策略。

要成功地完成软件开发，有一些基本的要求［7，8，42］：

（1）理解软件过程和产品；

（2）定义过程和产品质量；

（3）评估成功和失败；

（4）为项目控制提供反馈信息；

（5）从经验中学习；

（6）整理并重用相关经验。

无论是对工业界的、学术界的软件工程研究，还是寻求持续改进的学习型组织的软件工程研究，经验研究都是达到以上要求的重要方法。Basili 在提出质量改进范式（Quality Improvement Paradigm）［7］时同时提出的经验工厂（Experience Factory）就是学习型组织的一个范例，在本节的后续部分将进一步描述。该方法也包括一个通过度量来定义和评估一组操作目标的机制，即所谓的目标/问题/度量标准 GQM（Goal/Question/Metric）方法［17］，下文中将详细介绍。关于 GQM 方法更详细的介绍参见 van Solingen 和 Berghout［172］。

2.9.1 过程变化的经验评估

一个寻求改进的机构在引入过程变更（如，新的方法或工具）以改进其工作方式之前，通常希望对其影响进行评估。经验研究是获得变更影响的客观和量化信息的重要方法。在2.2～2.4节中，我们介绍了三种经验策略：调查法、案例研究和实验，2.5节中对这三种策略进行了比较。本节介绍在软件过程变更评估中如何使用这些策略［177］，意在得到一种恰当的方法来处理从研究到工业应用的技术转移。技术转移 24
和使用经验策略有关的技术转移步骤将在2.10节中讨论。

图2-1将各种策略放置于相应的研究环境中。该策略排序基于“常规”的研究规模。目的是通过对各研究中最常见的实施序列进行讨论，使研究结果能可控地付诸实践。调查法不会在很大程度上干预软件开发，因此风险较低。实验与实际项目相比通常规模有限，而案例研究往往针对特定的项目。此外，在进行工业案例研究之前，往往会在大学实验室进行实验以降低成本和风险，详见Linkman和Rombach的文章［113］。

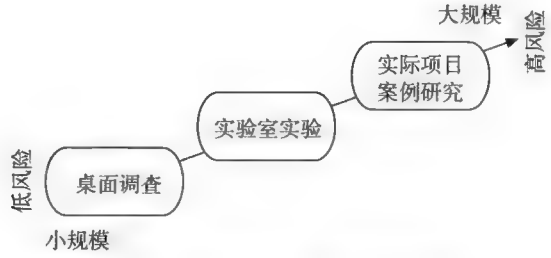


图2-1 调查法、实验和案例研究

有如下三类研究环境。

- 桌面：在未执行被变更过程的情况下对变更方案进行离线评估。因此，该类评估不牵涉使用方法和工具等的人员。在桌面环境，适宜采用调查法，如基于访谈的评估和文献调研等。
- 实验室：在离线的实验环境中（in vitro）[⊖]，对变更方案进行离线评估。在该环境配置下进行实验，且部分过程在受控方式下执行。
- 实际项目：在真实的开发场景中评估变更方案，即在在线的真实环境（in vivo）[⊜]进行观测，如试点项目等。在该环境下进行受控实验成本太高，通常采用案例研究更为恰当。

图2-1中，各研究环境按照项目规模和风险递增的顺序排列。例如，如果希望在真实环境中的一个大规模设计项目时尝试使用新设计方法，可以考虑先将该方法应用 25

⊖ 拉丁语“in the glass”，意指测试管中的化学实验。
⊜ 拉丁语“in life”，意指真实环境中的实验。

在一个作为试点研究的开发项目中。当然，这样做相对于桌面试验和实验室环境而言风险更大，因为如果过程变更失败，将危及交付产品的质量。案例研究和实验的成本通常高于桌面评估，因为桌面研究并不涉及开发过程的执行。需要指出的是，这里提到的成本是针对对同一件事情的调研而言的。例如，首先通过访谈的方法获得对新方法的预期影响很可能比做一个控制实验的成本要低；相应地，实验又比冒着采用新技术的风险在实际项目中采用新方法的成本要低。

在实际开发项目中开展案例研究之前，为了降低风险，需要在桌面或实验室环境中进行有限的前期研究。然而，对每一个变更方案在研究的顺序和成本方面，没有统一的结论，需要根据具体情况来评估哪种经验策略更高效，关键在于基于成本和风险选择最佳策略。在大多数情况下，我们建议研究从小规模开始，然后随着知识的积累、风险的降低，再增大研究规模。

除了考虑如何选择研究策略之外，还需要关于如何进行过程改进、如何搜集数据并存储信息等方法学的支持。这个问题将在下文逐一讨论。

2.9.2　质量改进范式

质量改进范式（QIP）［7］是一个专门为软件企业定制的改进方法。它类似于计划/执行/学习/处理循环 PDST（Plan/Do/Study/Act）［23，42］，包括如图 2-2 所示的六个步骤。

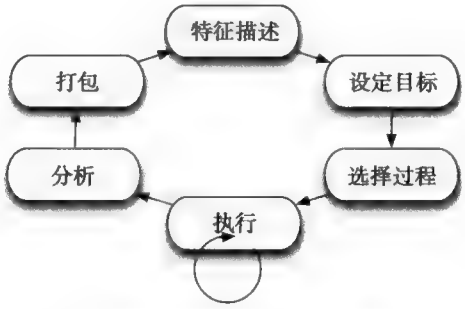


图 2-2　质量改进模型的六个步骤［7］

这些步骤的含义如下［16］。

（1）特征描述。基于可获取的模型、数据和直觉等理解环境。使用组织当前的业务过程建立基线并描述其关键特征；

（2）设定目标。根据初始特征描述和企业战略需求，设定可量化的项目成功目标、组织绩效和改进程度。应基于特性描述阶段建立的基线定义合理的期望目标。

（3）选择过程。根据环境的特征和设定的目标，选择合适的改进过程、支持方法和工具，确保它们和设定的目标一致。

（4）执行。完成产品开发，并收集关于目标达成的数据进行项目反馈。

（5）分析。在每个项目结束时，分析收集的数据和信息来评估当前实践、确定问

题、记录发现，为将来的项目提出改进建议。

（6）打包。将此次改进中新获得或更新的或精化后的经验、模型以及其他形式的结构化知识和以前项目中获得的进行整合。

QIP 中包括两个反馈环［16］，如图 2-2 所示。

- 项目反馈环（控制环，Control Cycle）是指在执行阶段向项目提交的反馈。无论组织的目标是什么，作为试点的项目都应该尽可能采用最佳的方式使用资源，因此建立项目和任务一级的定量指标是有利于预防和解决问题的。
- 企业反馈环（资本环，Capitalization Cycle）指需要提交给组织的反馈环。它有两个目的，其一是在项目结束时将项目数据与该组织的常规情况进行比较，提供关于该项目绩效的分析数据；其二是分析其一致性和差异性。该反馈有利于积累可重用经验，以便在其他项目中使用。

26
~
27

2.9.3 经验工厂

建立 QIP 的基础是：软件开发的改进需要持续学习。经验可以打包到经验模型中，以便于有效地理解和修改。这些经验模型存储在一个库中，称为经验库（Experience Base）。正在执行的项目可以访问、修改和重用这些模型。

QIP 强调将项目开发（由项目组织进行）与可重用经验的系统学习和打包（由经验工厂执行）进行逻辑分离［8］。经验工厂是通过分析和综合各类经验来对产品开发提供支持的独立组织，作为这些经验的仓库，在需要时为各种项目提供经验，见图 2-3。

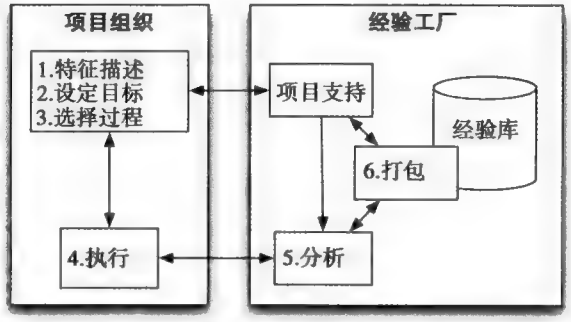


图 2-3 经验工厂

经验工厂以“通过人、文档以及自动化支持手段来构建不同过程、产品和其他形式知识的非形式化、形式化或系统化的模型和度量”来包装经验［16］。

项目组织的目标是生成和维护软件。项目组织为经验工厂提供项目和环境特征描述、开发数据、资源使用信息、质量记录和过程信息。它还提供由经验工厂提供并在该项目中使用的模型的实际效果的反馈信息。

经验工厂处理来自开发组织的信息，并将反馈信息，连同从类似项目剪裁得到的目标和模型一起直接返回给每个项目。它还为特定的项目提供基线、工具、经验教训

和数据。

为了改进，软件开发组织需要引入新技术。为此，需要做实验并记录开发项目取得的经验，从而最终改变当前的开发过程。当新技术与当前实践有本质性不同时，为降低风险可以采用离线评估。如上所述，变更评估可以采用控制实验（小规模的精细评估）或案例研究（研究尺度效应）方法。对这两种情况，下文描述的目标/问题/度
28 量方法提供了一个十分有用的框架。

2.9.4　目标/问题/度量方法

目标/问题/度量方法（GQM 方法）［17，26，172］基于以下假设：如果组织希望有目的地进行度量，必须遵循以下几点。

（1）为组织及其项目设定目标；

（2）将目标分解到数据，这些数据能够可操作地定义目标；

（3）提供一个框架，解释这些指征目标的数据。

应用 GQM 方法所得的结果是，针对一组特定问题的度量模型和一组解释度量数据的规则。

由此产生的度量模型有三个层次，层次结构如图 2-4 所示。

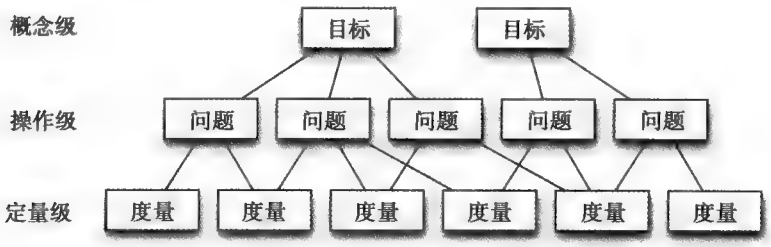


图 2-4　GQM 模型层次结构图

（1）概念级（目标）。为一个对象定义目标有着不同的原因，同时还要考虑不同的质量模型、不同的视角以及涉及的特定环境。度量对象是产品、过程和资源（也见第 3 章）。

（2）操作级（问题）。一组问题，用来描述如何基于一定的特征模型来评价某特定目标的达成度。问题描述度量对象（产品、过程和资源）的质量方面，并从所选定的视角确定其质量。

（3）定量级（度量）。与以上各问题相联系的一组数据，从而定量地回答这些问题（无论是客观还是主观）。

设定目标的过程是成功应用 GQM 方法的关键。制定目标一般基于以下三方面：组织的政策和战略，过程和产品的描述，组织模型。当目标被明确描述后，就能基于这
29 些目标得到问题。一旦问题形成，就可以将这些问题与适当的度量标准关联起来。

Briand 等［26］、van Solingen 和 Berghout［172］提出了将 GQM 方法应用于基于度

量的过程改进的实用指导。第3章将进一步对度量进行详细阐述。

2.10 基于经验的技术转移

经验研究不仅具有自身的价值，也是促进学术界和工业界进行知识交流与改进的最大推动力，例如上文讨论过的技术转移。软件工程属于应用研究领域，因此希望研究工业有关问题。在很多情况下，仅做学术研究是不够的，尤其是针对在工业界充满挑战性的领域，如需求工程、软件测试等。软件工程研究最好由学术界和工业界联合进行，以促进知识的双向转移，并最终实现新方法、新技术和工具从学术界到工业界的转移。联合研究提供了极好的机会，通过基于具体证据的研究来改善工业软件开发，因此是基于证据软件工程［48，100］的一个极佳案例。

Gorschek 等人基于长期的合作探索，提出了一个技术转移模型［66］。下文总结性地介绍该模型中的七个步骤，以展示在经验驱动的改进中如何使用不同的经验研究方法，尤其是实验的方法。该模型与2.9节中的软件过程改进密切相关，如图2-5所示。该模型的重点是使用不同的经验方法来为真实的工业问题创造性地给出解决方案，并实现到工业应用中。

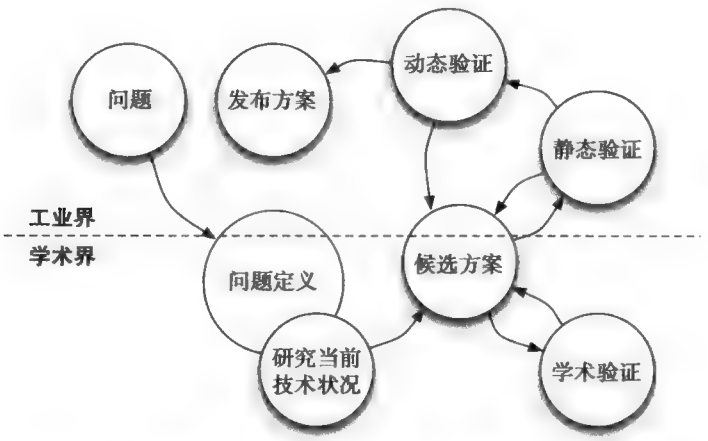


图2-5 技术转移模型（改编自 Gorschek 等人的描述［66］）

工业问题识别。第一步是识别工业界在某个特定上下文环境中面临的实际挑战，这意味着研究人员要到工业合作伙伴现场去。可以采用2.2节介绍的调查或访谈等方法来识别挑战。目的是找到挑战，尤其是适合研究的重要问题。所识别的挑战都必须能够被定义为研究问题，以避免研究人员仅以处理短期问题的顾问身份而告终。

这一步最大的收获是提供了一个双方建立信任的机会，并让工业伙伴及其雇员习惯有研究人员出现在其工作环境中。根据 Wohlin 等的说法［179］，在这一阶段来自管理者和其他参与合作工作的工程师的承诺是确保未来成功的关键。

问题定义。基于已识别的挑战，将这些挑战描述为存在的研究问题（research

problem），从而也就提出了需要解答的研究问题。如果识别了多个不同的挑战，则需要给出优先级。然后，给已经选定的挑战确定一个主要联络人。联络人最好不只是任命的，他应该是公司里的领导、合作研究的拥护者。这有助于联系到公司内合适的人员，以确保在需要的时候研究人员能访问系统、查看文件和数据。

研究人员开展文献检索是定义研究问题的必要部分。这可以采用第 4 章提出的系统文献综述方法。文献调研的目的是了解针对所识别的工业挑战目前已存在的解决方法。这有助于理解已有方法和实际工业需求之间的关系。

候选方案。基于已有的方法和实际需要，设计一个适合公司当前过程、方法、技术和工具的候选方案。该方案的设计应与工业伙伴紧密合作完成，以保证持续的可应用性。尽管给出的是针对一个公司的特定解决方案，但研究者的目的是开发一个通用解决方案，然后针对特定环境将其实例化。

30
~
31

学术验证。为了降低风险，方案的初步验证最好是在学术环境下进行，即离线验证。很多情况下可能是本书若干章中描述的实验或者一个学生项目的案例研究。第 5 章概要介绍案例研究方法。学术环境下的验证可以让学生或者工业伙伴代表作为主体来进行。

这一步的主要目的是发现建议方案中是否存在明显的缺陷，并对候选方案提出改进建议。学术环境下的验证是为了保证能将尽可能好的方案提交给工业界应用。

静态验证。静态验证是指工业代表离线评估候选解决方案。候选解决方案可以通过对不同工业代表（最好是不同的角色）进行访谈或者联合举行研讨会来进行验证。此外，最好在初期阶段就向企业简要介绍该候选解决方案，以便对方能够尽早了解该方案，同时也给对方一个在早期发表意见的机会。这也有助于减弱在将新方案集成到企业软件开发项目时可能出现的阻力。

基于静态验证的反馈，新解决方案可能需要做出调整。这七个步骤是迭代，因此这不仅仅是顺序执行问题，不应将其视作没有反馈环的瀑布方法。

动态验证。一旦新方案通过静态验证，并获得许可和承诺实施该方案，就要进行动态验证。最好采用试点评价的方法。具体如何进行动态验证取决于解决方案的类型。新解决方案可能会用于一个项目、子项目、部分系统或特定的活动。建议密切遵循动态验证，以评估解决方案。第 5 章将介绍如何采用案例研究方法进行动态验证。

发布方案。通用解决方案必须针对具体场景进行裁剪。需要确保任何研究方案都恰当地移交给了工业伙伴，且该公司能提供足够的支持，包括相关方面的说明、培训和潜在的工具支持。后者虽然不是研究人员的责任，但他们必须支持合作伙伴，以确保新解决方案的转移被适当地集成到组织中，之后才能开始对下一个挑战的研究。

通过案例研究可以对更广泛的使用情况进行经验研究，有助于为合作研究的新方案提供经验证据。

32

结束语。概括转移模型，以说明如何采用不同的经验策略将基于需求识别的研究成果转移到实际工业应用中。

最后，我们注意到，工业代表主要对针对其特定环境的解决方案感兴趣，而从研究者的角度来看，它只是通用解决方案的一个案例。因此，合作双发的主要关注点可能不同，但最后他们都受益于共同合作。工业合作伙伴获得了针对其某个挑战的解决方案，而研究人员能够在实际工业环境下评价其研究成果。Gorschek 和 Wohlin [65] 给出了一个需求抽象的通用解决方案案例，Gorschek 等人 [67] 给出了该方法的一个工业实例化方法。

2.11 实验中的伦理学

任何涉及将人作为主体的经验研究活动都必须考虑伦理方面的问题。某些方面受国家法律监管，而有些方面则根本不存在任何相关规定。Andrews 和 Pradhan 提出了软件工程领域的伦理问题，发现了现有政策的不足 [3]。Hall 和 Flynn 在英国调查人们的伦理实践和伦理意识时发现他们在这方面惊人地无知 [71]，而这绝非特例。

Singer 和 Vinson 引发了对伦理问题的讨论 [158]，并陆续讨论了伦理问题的案例 [159]，给出了经验研究的实践指南 [174]。主要有四条原则。

- 实验主体者必须知情并同意参与。这意味着他们在决定是否参与之前，有权访问该研究的所有相关信息。他们的决定必须是明确和自由的，不是隐含依赖于管理人员、教授等。
- 为了鼓励实验主体愿意接受经验研究的风险，研究必须具有科研价值，即便只具有很小的科研价值。
- 研究人员必须采取一切措施保证数据和敏感信息的保密性，即使这样做会与出版利益发生冲突。
- 综合考虑和权衡风险、损害以及收益。对受益的考虑必须充分，不能只考虑某些实验主体的利益，而应考虑整个实验主体群体和组织的利益。

以上准则被用在一个经验研究的计划制定、执行和报告撰写中时，往往被具体化为实践指南。下面我们给出了 Sieber [156] 中的一份实验主体在实验中可能面临的风险清单。

伦理审查。在某些国家，如加拿大、美国和澳大利亚等，法律要求对涉及人类实验主体的研究要进行伦理审查，必须遵循针对这类研究的规程和文件。这意味着需要 33
提交一份申请报告给大学或政府机构的伦理审查委员会（Ethical Review Board）进行审批。目前这些规程主要来自于生物医学研究的需求，因此通常不适用于软件工程研究。Vinson 和 Singer 提到，在加拿大，人们并不清楚使用源代码（由人编写并披露其信息）及其数据的研究是否应该遵循审查规程 [174]。

审查中所需的文档通常包含一份项目描述，包括实验主体和实验方案的细节，一份关于如何获得知情同意书的文档，以及该项目在伦理方面的审查结果。

知情同意。面向人的经验研究（如实验）的基础是实验主体是自愿参与的，他们有足够的信息来决定是否参与。并且，实验主体可以随时选择退出而不受到任何处罚。

为了使这一决策过程清晰明了，同意书应该以书面的形式呈现。

知情同意书通常包括下列要素［174］：

- 研究项目标题：用于了解研究动机。
- 联系信息：包括研究和伦理方面的联系信息。
- 同意和理解：实验主体陈述其理解并愿意接受项目条件。
- 退出：陈述退出且不受处罚的权利。
- 保密性：对数据处理和参与者信息保密的承诺。
- 风险与收益：明确列出实验主体面临的风险和获得的收益。
- 澄清：实验主体有权要求知晓他们在研究中扮演的角色。
- 签名：通常是实验主体和研究人员双方的签名；双方各执一份，以表示双方意见达成一致。

在某些实验设计中，完全公开实验目标和实验步骤可能会影响实验的执行效果。比如，如果预先知道实验假设，实验主体可能会在实验中相应地改变其行为。因此可以采取半公开的方式，即在更高的抽象层次上呈现实验目的和实验步骤。

对于在公司进行的经验实验（在线），必须同时由组织和各个实验主体签署同意书。特别要说明的是，不能命令实验主体参与，且他们可以自由退出而不受到处罚。另外，还必须考虑公司内的保密问题和敏感结果。

同意书还应说明它仅适用于当前研究目标，还是将来为了别的研究目标进行深入研究时仍然适用。

保密性。实验主体必须承诺对与研究人员分享的任何信息进行保密。保密性包括
34 三个方面［174］：

- 数据隐私，指对数据的访问限制，例如利用密码保护和加密。
- 数据匿名，指将实验主体的身份标识与数据分开。
- 匿名参与，意味着同意书是保密的。

由于经验研究（包括实验）旨在得出一般性结论，这与细节保密没有本质冲突。数据隐私问题也可以通过良好的工作实践来解决。然而，由于实验主体的数量较少，因此存在根据信息追踪到个人的风险，即使匿名也会存在这种风险。此外，对于研究的外部有效性（见第 8.7 节），应该报告研究的上下文信息，这可能会和匿名性冲突。

参与者的匿名性是最难实现的。对于实验招募的学生，尽管他们可能有权利拒绝参与实验，但很难不让研究者知道哪些学生参与了实验。同样在企业中，管理者很容易知道谁参与了这项研究。Vinson 和 Singer 的建议是“对于涉及学生的研究，研究人员应避免在课堂上招募学生和避免招募自己的学生”［174］，但该建议很少有人遵循。

敏感结果。经验研究的结果对于不同的利益相关者，其敏感的方面也是不同的。实验主体的个人表现就是一个例子，往往会是管理者或教授希望看到他的个人表现。经验研究的结论也可能是敏感的，尤其是与该项目投资方有利害关系时。如果一个实验不支持研究者的假设，该结果也可能对他们是敏感的。

下面这些情况强调利益相关人员的道德标准。可以采取多种独立性的措施进行约束，如：

- 实验主体，确保严格执行保密规程，独立地揭示事实（犯罪豁免［159］）；
- 投资方，在针对公司的知情同意书和研究项目合同中明确声明对于匿名结果的独立出版权利；
- 研究人员，最好让同行而不是实验人员进行匿名数据（包括主体和尺度）的独立统计分析，尤其当实验处置（treatment）是实验人员自己设计的时候。这也可以减少受到实验人员主观期望带来的威胁。

这些行动降低了陷入伦理困境的风险，并提高了所有经验研究的有效性。

诱因（Inducement）。为了招募实验受试者，必须有足够的诱因来吸引他们参与。 35
通过应用新方法获得经验和知识或许是足够的诱因。为了公平对待所有参加者，所有实验主体都应该有机会了解所有实验处置，即使实验设计并不要求如此。

可能会涉及将金钱作为诱因，例如，以现金支付、抽奖形式，或为职业实验主体支付薪水。无论以什么形式，诱因必须适宜于确保参与者是真正自愿参加的，并没有受过大的经济驱使或其他诱因所强制。

反馈。为了保持和研究实验主体长期的合作与信任关系，结果和分析的反馈是十分重要的。实验主体不必同意分析结论，但是应该享有获取实验相关信息和结果的权利。从保密性的角度，如果可能的话，个体绩效数据应该和总体分析结果一起报告返回。

基于伦理的结论。Singer 和 Vinson 在其早期工作中寻求制定经验软件工程的一套伦理行为准则［159］。然而，10 年过去了，软件行业仍没能开发出一套类似的规范，最接近的也就是上文总结的 Vinso 和 Singer 的指南。研究资助机构开始要求采用通用的伦理准则，虽然可能并不合适。凝练和制定适合于经验软件工程研究的伦理准则对于实验主体（保护对象）和该研究领域的发展都很有好处。

2.12 练习

2.1 定性与定量研究的区别是什么？

2.2 什么是调查法？请举例说明软件工程领域中不同类型的调查法。

2.3 实验重现和系统文献综述在构建经验知识中起到什么作用？

2.4 在技术转移方面，“经验工厂”如何将目标/问题/度量方法与经验研究相结合？

2.5 先观察后做实验的主要伦理原则有哪些？ 36

第3章
Experimentation in Software Engineering

度　量

软件度量是使得项目、产品和过程可控的关键因素。正如 DeMarco 所言："你不可能控制你无法度量的东西"［41］。此外，度量也是经验研究的中心环节。经验研究用于分析某些输入对被研究对象的影响。为了控制研究过程并看见效果，必须同时对输入和输出进行度量，以描述什么样的输入会对输出产生影响。没有度量，便不可能对研究进行操控，也就不能进行经验研究。

度量（Measurement）和度量值（Measure）的定义为［56］："度量是指按照清晰定义的规则为描述现实世界中实体的属性而为其分配数值或符号的过程。"度量值则指其中用于描述属性的数值或符号。

我们通过研究度量值来对实体进行评估，而不是直接进行评估。度量指标（Metric）这个词在软件工程中也很常见。它有两种不同的含义。首先，软件度量标准是一个表示软件工程度量领域的术语，如 Fenton 和 Pfleeger 的书中［56］对其的使用。其次，度量指标也用于表示被度量的实体。例如，代码行数（LOC）就是一种产品度量指标。更确切地说，它是程序规模的度量值。Shepperd 对软件度量有进一步的讨论［150］。

本章介绍基本度量原理。3.1 节介绍度量理论的基本概念及度量的不同尺度类型。

3.2 节介绍软件工程中的度量值实例及其与统计学分析的联系。3.3 节则在实践层面对度量进行讨论。

3.1　基本概念

度量值是从实体属性到度量所得值的一个映射，通常为一个数值。实体是指在现实世界中可以观察的对象。将属性映射为一个度量值的目的是形式化地描述和处理该属性。因此，度量值的一个基本特性是必须反映该属性的经验观测结果［57］。也即，如果观察到对象 A 比对象 B 长，则 A 的度量值必须大于 B 的度量值。

当我们在经验研究中使用度量值时，首先要确保它的有效性。要确保有效，度量值不能违背其度量的属性的任何必要特性，并且必须是该属性一个恰当的数学表征。

一个有效度量值可区分不同的对象，但在度量误差范围内，对象可以具有相同的度量值。度量值也必须反映我们对属性的直觉概念，及对象间相互区分的方式［97］。度量值必须具有分析有效性和经验有效性。度量值的分析有效性是指能否准确可靠地捕获所关注的内容；经验有效性（有时亦称统计或预测能力）则描述该值与其在不同环境条件下度量结果的相关性。

效应量（Effect size）是量化两组对象之间差异的一种简单方式。它在实验中非常重要，虽然不一定具有现实意义，但它可能会显示两组对象在统计上的显著差异。通常认为，如果实验中主体数量足够多，即可显示出统计上的显著差异，但这并不意味着有现实意义。有时可能因为差异太小，或是因为发现差异的成本太高。

将属性映射到度量值有多种不同的方式，我们将其称为尺度（Scale）。如果属性是对象的长度，我们可以用米、厘米或英寸来度量它，这里米、厘米、英寸是度量长度的不同尺度。

由于对同一属性的度量可以用多个尺度，所以有时希望将度量从一个尺度变换到另一个尺度。如果这个变换保留了对象间的关系，则称其为容许变换（Admissible Transformation）[56]，也称尺度改变（Rescaling）。

用属性的度量值可对对象及不同对象之间的关系进行陈述。如果尺度改变不影响陈述的正确性，则称该陈述是有意义的，否则就是无意义[27]。例如，如果分别度量对象A和B的长度，得A为1m，B为2m，我们就可以声称“B的长度是A的两倍”。如果用厘米或英寸作为尺度去度量，这个陈述仍然是正确的，因而它就是有意义的。 38
又如，如果度量房间A和房间B内的温度，分别为10℃和20℃，则有陈述“房间B比房间A温暖一倍”。但如果用华氏温标去重新度量，则将得到50℉和68℉，之前的陈述不再正确，因而这个陈述就是无意义的。

根据不同尺度之间可否进行容许变换，可将尺度进行分类。具有相同特性的尺度属于同一尺度类型，不同的尺度类型有强弱之分：可表达的有意义陈述越多，则尺度类型越强大。下面将介绍一些常用的尺度类型。

度量值还有另两种分类方式：（1）直接度量值还是间接度量值；（2）客观度量值还是主观度量值。这两种分类方式也将在后面进行讨论。

3.1.1 尺度类型

最常见的尺度类型有如下四种[27, 56, 57]⊖。

（1）定类：定类尺度（Nominal Scale）是尺度类型中最弱的一种。它只将实体属性映射为一个名字或符号。这种映射可视作依据该属性对实体的一种分类。

定类尺度进行容许变换的前提是实体必须一一映射。

定类尺度的例子有：分类、标记和缺陷分类。

（2）定序：定序尺度（Ordinal Scale）将实体按某排序标准进行排序，因而比定类尺度更强。排序标准如：“大于”“优于”和“更复杂”。

定序尺度进行容许变换的前提是保证实体的顺序不变。例如，若M′和M是同一属性的不同度量值，F是单调增长的函数，则M′ = F(M)。

定序尺度的例子：年级、软件复杂度。

⊖ Fenton等[56, 57]提出了第五种尺度类型，即绝对尺度。这是定比尺度的一个特例，主要指值本身仅用于转移意义。计数是绝对尺度的一个例子。

（3）定距：当关注点是两个度量值的差，而非度量值本身时，通常使用定距尺度（Interval Scale）。该尺度类型和定序尺度一样会将实体进行排序，但不同的是这里考虑
39 两个实体间的“相对距离”。因此该尺度比定序尺度类型更强。

定距尺度进行容许变换的前提是度量值互为线性组合，例如，M′和 M 是同一属性的不同度量值，$M' = \alpha M + \beta$。定距尺度度量值在软件工程中较为少见。

定距尺度的例子：用华氏温标或摄氏温标度量温度。

（4）定比：当度量值存在一个有意义的零值，并且两个度量值间的比率也有意义时，可以使用定比尺度（Ratio Scale）。

定比尺度进行容许变换的前提是它们有相同的零值，并且度量值间的区别仅在于常数倍的不同。例如，M′和 M 是同一属性的不同度量值，$M' = \alpha M$。

定比尺度的例子：长度、绝对温标、某一开发阶段的持续时间。

度量尺度与定性和定量研究有关。进一步来说，还与度量值可使用的统计方法有关。详见第 10 章。据 Kachigan 的研究［90］指出，定性研究一般使用定类尺度和定序尺度；定量研究一般使用定距尺度和定比尺度。

3.1.2　客观和主观度量

有时对属性的度量必须考虑度量的视角。为此又将度量值分为如下两类。

（1）客观：客观度量值在度量中不需进行人为判断，因而只与被度量的对象有关。客观度量值可以由不同的研究人员多次度量，每一次得出的值在度量误差范围内都是一样的。客观度量值的例子：代码行数（LOC）、交货日期。

（2）主观：主观度量值与客观度量值相反。需要人做出某些判断来获得度量结果。因而，该度量值不仅与被度量的对象有关，还与人采用的视角有关。主观度量值在多次进行时每一次的结果都可能是不一样的。主观度量值一般使用定类或定序尺度类型。主观度量值的例子有：人员技能、可用性。

40 主观度量值总是会受潜在的偏见的影响，3.3 节将对此再进行讨论。

3.1.3　直接和间接度量

我们感兴趣的属性有时不能直接进行度量。对它们的度量值必须通过其他直接可度量的度量值派生。为了区分直接度量所得的度量值和派生的度量值，我们把度量值划分为直接度量值和间接度量值。

（1）直接：属性的直接度量是指可直接测量的，不涉及对其他属性的度量。直接度量值的例子有：代码行数、测试中发现的缺陷数量。

（2）间接：属性的间接度量涉及对其他属性的度量。间接度量值是由其他度量值派生出来的。间接度量值的例子有：缺陷密度（缺陷的数量除以代码行数）、程序员生产率（代码行数除以程序员的工作量）。

3.2 软件工程中的度量

在软件工程中，我们关注的对象主要可以分为三类。

- 过程（Process）：过程描述了生产软件所需的活动。
- 产品（Product）：产品是过程活动中产生的制品、可交付物或文档。
- 资源（Resources）：资源是过程活动所需的对象，如人员、硬件或软件。

在上述各类中，我们也区分内部和外部属性［55］。内部属性是指仅仅根据对象本身就可以被度量的属性。外部属性的度量则必须考虑与其他对象之间的关系才能进行。不同软件度量值的例子在表3-1中列出。

表3-1 软件工程中的度量实例

类 别	对象举例	属性类型	度量值举例
过程	测试	内部	工作量
		外部	成本
产品	代码	内部	大小
		外部	可靠性
资源	人员	内部	年龄
		外部	生产力

在软件工程中，软件工程师通常希望陈述一个对象的外部属性。但外部属性大多数是间接度量值，必须由该对象的内部属性派生得到。内部属性则大多是直接度量值。

度量值通常是度量程序的一部分。例如，Grady 和 Caswell［68］以及 Hetzel［75］曾讨论过构建软件度量程序的方法。

软件工程中的度量不同于其他领域的度量，如物理学。在后者，度量什么属性以及如何度量通常是很明确的。然而，在软件工程领域，有时很难以一种大家都认可的度量方式来定义属性［56］。另一个不同之处是，在软件工程中很难证明这些度量值是否都只是定类或定序尺度类型。间接度量值的验证往往更加困难，因为它不仅需要验证它所包含的直接度量值，还需要验证派生出外部度量值的模型。

当进行经验研究时，我们关注度量值的尺度类型，也关注其上可采用的统计分析方法。形式上，统计分析方法取决于尺度类型，但这些方法通常比尺度类型有更好的鲁棒性。基本的规则是，使用的尺度类型越强大，则可用的分析方法也越强大，参见第10章。

在软件工程中，许多度量值都常采用定类尺度或定序尺度来度量，虽然并不能证明它们不能用更强的尺度类型，但却意味着我们不能使用那些要求定距或定比尺度的更强的统计分析方法来开展经验研究。

Briand 等人［27］认为，即便不能证明我们能够使用定距或定比尺度，我们依然

可以使用更强大的统计分析方法。在失真不是很极端时，大多数强大的统计方法对于定距尺度的非线性失真仍然是鲁棒的。如果我们能谨慎地考虑风险，便可以使用更强大的统计方法来获得结果，当然前提条件是获得大规模的度量值样本。

3.3　实践中的度量

实践中，由研究人员定义度量标准，然后在经验研究的操作阶段收集数据。接下来考虑如何收集数据，最好能不要求实验主体花费太多的精力。在很多实验中，实验主体通过填写表格来提供数据；当然也可以使用工具来自动收集数据，比如使用开发环境。Lethbridge 等［111］讨论了一些通用数据收集技术。

41
|
42

由于收集到的度量数据是进一步分析的基础，所以它们的质量对于研究的后续分析至关重要。这也就意味着真正弄清楚应该收集哪些类型的度量数据及其属于哪一尺度类型是很重要的。同样，弄清楚它们服从何种概率分布也很重要，尤其是要知道是否满足正态分布。

当考虑到概率分布时，就可以使用描述性统计方法来调研。比如，可以在图中标绘数据，或者采用其他分析技术来分析该数据在何种程度上呈正态分布。详细介绍见第 10 章。采用何种尺度类型取决于度量标准的定义，因此研究人员在定义度量标准时要对尺度类型有充分的理解。

度量标准的定义将在很大程度上影响度量数据的展示效果，而这正是研究人员所关心的。例如，Kitchenham 等人［102］比较了两种展示生产率的方法，结果表明，展示工作量和规模的散点图比展示随时间变化的生产率图表效果更好。一条普适的经验是，最好不要使用由两个独立度量值的比值构成的度量标准，除非你真的能说清楚该度量值的含义

在研究的操作过程中，确保所收集数据的正确性是非常重要的。这意味着，研究者需要在实验中采取质量保证措施。比如审查实验主体是如何填写表格的、检查数据的一致性等。关于数据验证的内容将在第 8 章讨论。

还有一个需要考虑的因素是，谁是实验中被调查内容的提出者或拥有者。正如 Kitchenham 等人建议的［98］，理想的做法是由新方法的提出者之外的其他人通过实验或其他研究方法来评价该方法。方法的提出者自然希望这个方法表现优异，其作为研究者时会有意或无意地选择对该方法有利的度量标准，因而存在风险。如果实验主体得知研究者就是被评估方法的提出者，也会影响其行为。如果实验要在新方法被提出的地方进行，那么度量标准的设计和选取应该经过外部研究人员的审查。

3.4　练习

3.1　什么是度量值、度量和度量标准，它们之间有什么关系？

3.2　四个主要的度量尺度类型分别是什么？

3.3 直接度量值和间接度量值的区别是什么?

3.4 软件工程中的度量分成哪三类?

3.5 什么是内部和外部属性?它们与直接度量值和间接度量值之间的常见关系是什么?

第4章
Experimentation in Software Engineering

系统文献综述

系统文献综述（Systematic Literature Reviews）旨在“定义、分析和解释与特定研究问题相关的所有可用的证据”[96]。其目标是对当前的事实给出一个完整、全面和有效的描述，因此“定义、分析和解释”必须以科学且严谨的方式进行。为了实现这个目标，Kitchenham 和 Charters 主要借鉴来自医学领域中的系统文献综述方法，对其进行评估[24]和适应性修改[96]，制定了适合软件工程领域的系统文献综述指南。该指南将其结构化为制定综述计划、执行综述和撰写综述报告三个步骤，本节将逐一阐述。

4.1 制定综述计划

制定系统文献综述计划包含以下几个步骤。

识别综述需求：对系统综述的需求主要来源于研究人员希望了解一个领域的最新进展，或者从业人员希望在制定战略决策或活动改进时使用经验性证据。如果某领域已或多或少地有一些系统文献综述，则应根据其所调研的范围和质量来评估是否满足当前综述的需求。系统文献综述是一种进行文献综述的研究方法。

指定研究问题：为了识别原始研究，以及从这些研究和分析中提取数据，需要指定系统综述的范围和聚焦研究的问题。因此，研究问题的提出必须经过深思熟虑，描述时需要精心措辞。描述研究问题时应考虑以下几个方面[96]。

- 证据采集的总体。即，综述对哪些人群、项目或业务感兴趣。
- 经验研究中的介入项（intervention）。即，被研究的技术、工具或方法。
- 对各种介入项的比较。即，控制处置是如何定义的？特别需要注意“无效对照组”介入项。因为在软件工程领域，“不使用该介入项”通常是种无效行为。
- 实验的结果不仅应该满足统计学意义，而且也要有实践意义。例如，如果需要用两倍的时间来获得某部分 10% 的改进，则实际意义不大。
- 务必要定义研究的背景，这是对总体的扩展描述，包括是在学术界，还是在工业界进行，具体工业部门，以及主体的动机[78，132]。
- 需要定义研究问题中的实验设计。

Staples 和 Niazi 建议由清晰聚焦的研究问题来界定系统文献综述的范围，以避免研究变得不可控[166]。

制定综述方案：综述方案决定了系统综述的流程。它也在综述的进行中扮演日志的角色。因此，综述方案是个“活”文档。Kitchenham 和 Charters 提出在综述方案中必须要覆盖以下几项内容[96]：

- 研究背景和意义；
- 研究问题；
- 针对原始研究的搜索策略；
- 文献的选择标准；
- 文献的选择程序；
- 文献质量评估检查单和评估过程；
- 数据提取策略；
- 提取数据的综合分析；
- 发布策略；
- 项目进度安排。

为了保证方案的一致性和有效性，最好进行同行评审。系统文献综述的经验表明，预先评审有助于确定研究问题的范围，以及在方案制定过程中随着被研究的问题逐渐清晰可以对研究问题进行调整［24］。

4.2 实施综述

46

实施综述就是将综述方案付诸实践。这包含以下几点。

研究的识别：该步的主要任务是指定搜索字符串并将其用于数据库搜索。当然也包括对期刊和会议论文集的手动搜索，以及对研究人员个人网站的搜索和通过发送问卷给研究者的搜索。基于参考文献前向或后向地系统搜索原始研究的方法叫做“滚雪球”［145］。

搜索策略是权衡的结果，目的是找到所有原始研究，同时不会导致很高的误报率（误报必须通过手工剔除）［43］。所谓误报（false positive）是指错误地将不正确的答案作为正确的结果输出。在这里，是指某篇文章由于能被检索到而被认为与当前主题相关，但是之后发现并非如此，所以应将其剔除。搜索字符串是根据所覆盖的领域和研究问题决定的。为了找到所有相关文献，必须搜索多个数据库，因此会重复找到同一文献，为此还需要识别并剔除这些重复的文献。最后，所找到的论文必须是某个主题所有论文中的一个样本。关键点在于这个样本确实是来源于预期的总体。

已发表的原始研究会存在*发表偏倚*，也即（在某种程度上）*正面结果比负面结果*更可能被发表。因此，还应搜索灰色文献，如技术报告、学位论文、被拒绝的出版物，以及正在进行的工作等［96］。

搜索结果和搜索动作日志都应该存储下来，可以采用文献管理系统来记录。

原始研究的选择：选择原始研究的核心在于选择的标准。为了避免偏倚，标准应该提前制定。然而，由于在计划阶段可能对涉及选择的各个方面考虑不周全，因此在选择过程中也可以对标准进行适当调整。

根据选择标准，识别出候选研究的集合。对于一些研究，只读标题和摘要就足以判断了，但对于某些研究则需要更全面的分析，例如根据论文所采用的研究方法或者

结论来判断是否选择。结构化摘要［30］可能对选择过程有帮助。

文献选择实际上是一个判断过程，且应基于定义良好的判断标准，因此建议至少两个研究者对每一篇论文进行评估，或者至少对随机论文样本进行评估。可以使用Cohen Kappa统计来度量评分之间的一致性度［36］，并将其作为系统文献综述报告中质量评估的一部分。然而，因为很多自动搜索出的论文很容易在研究者人工分析时被排除，所以注意应该要用一个相对来说较高的Cohen Kappa统计量。因此，分步骤地进行评估可能很重要，即首先移除那些明显不相关的论文，即使它们被搜索出来了。

研究质量评估：评估原始研究的质量非常重要，尤其是在这些研究报告的结果互相矛盾时。原始研究的质量可用来分析产生矛盾结果的原因或者在整合结果的时候用来权衡每个研究的重要性。

47

“研究质量”没有一个普遍认可并且实用的定义。医学中制订质量标准的尝试并没能映射到软件工程研究的质量范围中［47］。

最实用且有效的质量评估方法就是检查单（checklist），虽然其经验基础比较薄弱。Kitchenham等的一项研究表明，一个有效的评估至少需要三个评审人员［105］。用于经验研究质量评估的检查单也适用于经验软件工程文献的质量评估［96，105，145］。

如果将研究质量作为文献选择标准的一部分，那么可能会排除一些原始研究。值得注意的是，被评估的是原始研究的质量而不是报告的质量。但是，如果报告写得不好，会使该研究的质量很难评判。可能需要与作者联系来发现或者明确报告中缺失的信息。

数据的提取与监控：一旦确定了原始研究检查单，就需要提取原始研究中的数据了。需要设计一个数据提取表来从原始研究报告中收集所需的信息。如果在研究选择中采用了质量评估数据，则该数据提取表应分为两个部分：一个用于在质量评估过程中填写质量数据，另一个在数据提取过程中填写研究数据。

数据提取表是根据研究问题设计的。对于纯粹的元分析综合而言，数据是一组数值，表示主体的数量、对象的特征、处置的效果、置信区间等。对于弱同质的研究组，必定会包括更多原始研究的定性描述。除了原始数据之外，每个原始研究的评审人姓名、数据提取日期以及出版细节都应记录下来。

在原始研究中全面使用数据提取表之前，应该对其进行试点性使用。为了评估提取过程的质量，如果有可能的话应该由两位研究者，至少对研究中的样本，独立地进行数据提取。

如果一个原始研究不止在一篇论文中发表，例如一篇会议论文被扩展为一篇期刊论文，则这两篇文章只能算作一个原始研究。如果这两个版本都在数据提取中，大多数情况下更倾向于采用期刊论文，因为它更为完整。技术报告或者与作者的沟通也可作为数据提取源。

数据综合分析：最先进的数据综合分析方法是元分析（Meta-Analysis）。这里指用

于分析多个独立研究结果的统计方法。元分析假设被综合分析的研究应该是同质的（或同类型的），或者其非同质的原因是众所周知的［135］。元分析对比效应量和P值来评估综合的结果。基于对同质性的要求，它主要适用于重现试验。总之，元分析中所包含的研究必须满足以下几点［135］： 48

- 类型相同，例如，正式实验；
- 检验假设相同；
- 处置的度量和效应结构相同；
- 报告的解释性因素相同。

元分析主要包括以下三个步骤［135］：

（1）确定元分析包含哪些研究。

（2）从原始研究报告中提取效应量，或者在没有公布效应量时对其进行估算。

（3）结合这些原始研究的效应量来评测联合效应。

除了上面提到的原始研究的选择规程之外，元分析还应包括对发表偏倚的分析。如图4-1所示的倒漏斗图，这里观察到的效应量是对照研究规模的度量值来绘制的，例如方差的倒数或者其他离散度量值（参见10.1.2部分）。当一组原始研究完成时，其数据点分布在一个漏斗形状周围。漏斗中的裂缝表明有些研究还没有发表或者没有被找到。

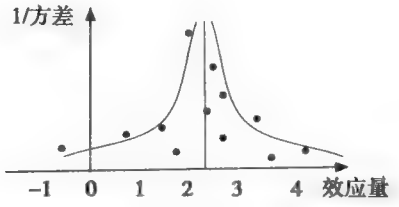


图4-1 12个假设研究的倒漏斗图示例

效应量是一个指示器，独立于各原始研究中所用的单位或尺度。它依赖于研究类型，但通常采用每个处置的平均值之差。这个度量值必须标准化以便于不同尺度之间的比较，也就是说要除以合并标准差［135］。

由于该分析假设各研究是同质的，因此可以采用一个固定的效应模型。元分析通过计算每个研究的效应量的均值来估测真实的效应量。为了保证模型条件得到满足，应采用如Q检验和似然比检验等检验方法来识别异质性。

对于异质数据，可以采用随机效应模型，允许存在未知因素导致的变化，该变化会影响原始研究的效应量。该模型提供了两种估算，一种是抽样误差估算，这和固定效应模型一样；一种是对异质子总体中变化的估算。 49

数据综合分析的弱形式化方法有描述性或叙述性综合分析。这些方法以更加突出研究问题的方式来表格化原始研究中的数据。作为表格化数据的最低要求，Kitchenham和Charters提出了以下几点［96］：

- 每个干预的样本大小；
- 每个干预的效应量估算及每个效应的标准误差；
- 每个干预平均值之差，以及这个差值的置信区间；
- 度量效应的单位。

统计结果可以用森林图来展示。森林图可以展现每个研究的处置之差的均值和方差。图 4-2 为一个森林图的例子。

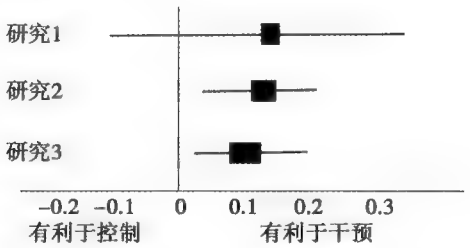


图 4-2　三个假设研究的森林图示例

综合分析非同质研究和混合方法研究需要定性方法。Cruzes 和 Dybå［39］调查了软件工程中的二次研究，包括经验证据的综合分析。他们找到了多种综合分析方法，很多来源于医学并且其中有七种方法已用于软件工程。下面简要介绍这些方法。详细内容参见 Cruzes 和 Dybå［39］及相关文献。

- 专题分析（Thematic analysis）是一种旨在从原始研究中识别、分析和报告模式或主题的方法。它至少应组织和呈现出数据丰富的细节，并解释被研究主题的各个方面。
- 叙述性综合分析（Narrative synthesis），上文已提到该方法，它讲述了一个源于原始证据的“故事”。它采用诸如数据表格化、分组和聚类、点票方法等作为描述性工具，将原始证据和解释结构化。叙述性综合分析适用于有定性或定量数据，或二者兼备的研究。
- 比较分析（Comparative analysis）旨在分析复杂的因果关系。它用布尔逻辑来解释原始研究中的因果关系。分析列出每个原始研究中的必要和充分条件，并且根据每个研究中独立变量的存在/不存在情况得出结论。这类似于 Noblit 和 Hare［127］的参数综合分析方法，参见 Kitchenham 和 Charters［96］。

50

- 案例调查（case survey）最初为案例研究而定义，但也可适用于非同质性的实验。它通过将具体问题的调查工具应用于每一个原始研究来聚合已有研究［114］，这类似于上文提到的数据提取。调查获得的是量化数据，因此聚合可以采用统计方法进行［108］。
- 元人种学（Meta-ethnography）将研究进行相互转换，然后将转换综合为超越个体研究的概念。元人种学研究将原始研究中的解释和说明作为输入数据。这类似于 Noblit 和 Hare［127］的“相互转换”和“驳斥性综合”，参见 Kitchen-

ham 和 Charters［96］。

- 元分析（Meta-analysis），如之前所提到，元分析是基于统计方法集成来自多个案例的定量数据。
- 概括分析（Scoping analysis）旨在给出某个领域研究的一个概貌，而不是去综合分析各研究成果。概括分析也被称为映射研究，详情参见 4.4 节。

不同于综合分析方法，敏感性分析主要分析不同研究子集的结果是否一致。比如，研究子集可以是只考虑高质量的原始研究，特定类型的原始研究，或者有良好研究报告（给出所有需要的细节）的原始研究。

4.3　撰写综述报告

和其他经验型的研究一样，系统文献综述也需要撰写报告给不同的读者。尤其是当综述的目的是为了影响从业人员时，报告的格式必须适合读者的需要。Kitchenham 和 Charters［96］列出了传播给目标从业人员的以下几种方式。

（1）面向从业人员的期刊和杂志；

（2）大众新闻稿和专业媒体；

（3）简短摘要的传单；

（4）海报；

（5）网页；

（6）直接与受影响对象沟通。

对于学术读者，对研究规程的详细报告是评估和评价该系统文献综述质量的关键。理想情况下，报告应包含研究方案的变化、被选入和被排除的原始研究的完整列表、分类数据，以及源于每个原始研究的原始数据。如果由于篇幅限制无法发表所有的细节，建议在线发布一个支持技术报告。Kitchenham 和 Charters［96］提出了一个学术报告的详细结构。

51

4.4　映射研究

如果文献综述的研究问题较为宽泛或者对该研究领域的探讨较少，则应采用映射研究而不是系统文献综述。映射研究［131］有时也称为概括研究［96］，它为某类研究搜索更广泛的领域，以得到某主题的当前技术发展水平或实践水平的概况。

映射研究与系统文献综述遵从同样的原则性过程，但是对文献的选择以及文献质量评估有不同的标准。由于其范围更广、研究类型更多样，与系统文献综述相比收集的数据与综合往往更趋于定性化。然而，对于映射研究的贡献和相关性来说，重要的是分析超出了单纯的描述性统计，而是将趋势和观察与现实世界的需求相结合。

Kitchenham 等人［106］总结了一个映射研究与系统文献综述的主要特征对比表，见表 4-1。

表 4-1　映射研究和系统文献综述的区别，根据 Kitchenham 等［106］制作

SLR 元素	系统映射研究	系统文献综述
目标	针对软件工程选题的文献分类和专题分析	聚合来自比较研究的信息，识别针对特定规程、技术、方法或工具的最佳实践
研究问题	通用的——与研究趋势有关。 形式有：哪些研究者、多少活动、什么类型的研究等。	具体的——与经验研究的结果有关。形式有：技术/方法 A 是否比 B 好？
检索过程	由主题领域定义	由研究问题定义，研究问题确定被调研的具体技术
辖域	广泛——包括所有与主题领域相关的论文，但只收集对论文的分类数据	专注——只包括与特定研究问题相关的经验论文。从每篇论文中提取其研究成果的详细信息
检索策略要求	通常不严格，只关心研究趋势。比如作者可以只检索一组目标出版物，可以限制为期刊论文，或 1～2 个数字图书馆	非常严格——所有相关研究都应该被找到。通常，系统文献综述小组不仅要检索数据源，还需要其他技术，如查看原始研究的参考文献、接触该领域的研究人员来了解他们是否在该领域开展新的研究
质量评价	非必须。理论研究和各类经验研究的复杂性导致评估很复杂	很重要，要确保结果是基于高质量的证据
结果	对一组关于某主题领域的论文进行不同维度的分类，以及对各分类中论文的数量进行计数	聚合所有原始研究的成果来回答特定的研究问题，可能带有限制（比如结果只适用于初学者）

4.5　综述举例

Kitchenham 等人报告软件工程领域在 2004～2008 年间发表了 53 篇系统文献综述［103，104］。他们得出的结论是，系统文献综述的发表数量在增长，并且综述的质量也呈上升趋势；同时也发现，在意识到并使用系统指南来撰写的综述和没有考虑参考任何指南撰写的综述之间仍然存在很大的差别。

在这些系统文献综述中，Sjøberg 等人［161］调查了在软件工程领域进行的实验研究。他们检索了从 1993～2002 年这十年间的九个期刊和三个会议集，浏览了 5453 篇文章找到了 103 个实验，即 1.9% 的论文做了实验。根据 Glass 等人的框架［63］，进行实验最频繁的两个研究类别是软件生命周期/工程（49%）和方法/技术（32%）。这是因为大量实验分别集中在审查技术和面向对象的设计技术上。

运用同一组原始研究，Dybå 等人［49］综述了软件工程实验中的统计能力，Hannay 等人［72］综述了软件工程中理论的应用。Dieste 等人［43］调查了同一组研究中不同的搜索策略，比如是否应该搜索标题、摘要或全文，以及应该搜索哪些相关的文献数据库。

Hayes［74］和 Miller［121］较早地尝试综合分析了五个关于审查技术的实验，

结果表明该领域的软件工程实验在统计元分析方面并不是足够同质的。他们还认为对元分析而言，原始数据必须可用，同样来源于原始研究作者的其他未发布信息也应公开提供。

在最近的一个关于结对编程有效性的文献综述里，Hannay 等人［73］对 18 个原始研究中的数据进行了元分析。他们报告了针对结果的三个组成部分：质量、时间和工作量的独立分析情况，并使用森林图可视化了该结果。

4.6 练习

4.1 系统文献综述和通用文献综述的区别是什么？

4.2 针对原始研究的检索策略有哪些？

4.3 为什么两位研究者在进行系统文献综述时要采用一些相同的步骤？

4.4 在原始研究上进行元分析应具备什么样的要求？

4.5 系统文献研究和映射研究的核心差别是什么？

52
54

第5章
Experimentation in Software Engineering

案例研究

“案例研究”这个术语在软件工程领域论文的标题和摘要中很常见。现在很多研究，从雄心勃勃、组织良好的实地研究，到小如玩具的示例都号称为案例研究。但后者更宜称为示例。对研究的分类也存在不同的分类体系，通常认为案例研究与实地研究、观察性研究等术语相近，关注的是研究方法学的特定方面。如，Lethbridge 等人将实地研究作为更通用的术语［111］，Easterbrook 等人则称案例研究是“五大研究方法”之一［50］。Zelkowitz 和 Wallace 为了区别于其他领域已有的术语，将项目监控、案例研究和实地研究都归为观察法［181］。当我们试图汇集多种经验研究方法时，这些过多的术语容易引起混淆。

很多类型的软件工程研究，只要被研究对象是当前存在的且难以孤立出来进行研究，都可以采用案例研究的方法。案例研究不会产生和受控实验一样的结果，如因果关系等，但它可以让我们对真实环境中的现象有更深入的理解。正因为与分析型、受控的经验研究不同，案例研究总被批评为价值小、无法泛化、带有研究者偏颇性等。但如果在实践中能采用合适的研究方法，并接受“知识不仅是统计学意义的”的观点，则上述批判就不是问题了［59，109］。

本章的目的是为实施案例研究的研究者提供一些指南。本章内容基于 Runeson 和 Höst［145］的研究，更多关于软件工程领域案例研究的阐述可以查阅 Runeson 等人的研究［146］。具体来讲，我们通过对现有检查单［79，145］进行系统化分析，给出一个新的检查单，之后由博士生和国际软件工程研究网络（International Software Engineering Research Network）组织的成员进行评估，并进行相应的更新。

本章对于什么是软件工程中“好”的案例研究没有提供绝对的定义，而是更关注一系列有助于提高研究质量的话题。对每个话题的最低要求是根据其使用环境来决定的，并且很可能随着时间的推移而变化。

本章的主要内容如下。首先，5.1 节介绍了案例研究的使用环境，讨论了在软件工程中使用案例研究的动机，并定义了案例研究过程。5.2 节讨论了案例研究的设计和数据收集的计划。5.3 节介绍了数据收集的具体过程。5.4 节讨论数据分析，所得结果的报告在 5.5 节中讨论。

5.1 案例研究的使用环境

案例研究的三个最常见的定义分别是由 Robson［144］、Yin［180］、Benbasat［22］等人提出的。这三个定义一致认为，案例研究是针对真实环境中当前存在的某种现象进行调查研究的一种经验方法。Robson 称其为一种研究策略，并强调了分析时

对多种来源证据的使用；Yin 表示案例研究是一种调查，并指出被研究现象与其存在环境之间的边界是不清晰的；而 Benbasat 等人的定义更加具体，认为信息是从少量实体（人，群体，组织）收集得到，并且缺乏实验控制。

行动研究与案例研究密切相关，因为前者的目的是“影响或改变研究所关注的某些方面”［144］。更严格地讲，案例研究是纯观察性的，而行动研究关注并参与变化过程。在软件过程改进［44，85］和技术转移研究［66］中，如果研究者积极参与改进过程，那么研究方法就是行动研究。不过，在研究变化带来的影响时，例如进行事前分析和事后分析时，我们将用到的方法归为案例研究。在信息系统研究中，行动研究十分常用，因此也就有了关于如何寻找行动和研究间平衡的探讨，参见 Baskerville 和 Wood－Harper［21］或 Avison 等人［5］的研究。对于行动研究的研究部分，案例研究的指南同样适用。

Easterbrook 等人［50］把民族学研究也作为主流研究方法之一。我们更倾向于将民族学研究视作一种关注于文化习俗的特殊案例研究［50］，或者依托于大量参与者－观察者数据的长期持续性研究［98］。Zelkowitz 和 Wallace 定义了软件工程领域的“四大观察法”［181］：项目监控、案例研究、断言和实地研究。我们倾向于将项目监控视为案例研究的一部分，将实地研究视为多案例研究，而断言则不是一个公认的研究方法。

Robson 总结出的观点“许多柔性的设计研究，虽然没有明确标示，但同样可以有效地视为案例研究”［144］，在软件工程领域内同样适用。 56

案例研究可能包含其他研究方法的一些元素，例如，案例研究中可能会采用调查的方法，文件检索往往是案例研究的先行工作，档案分析也可能是案例研究中数据收集过程的一部分。一些民族学研究的方法，例如访谈、观察等，也常在案例研究的数据收集阶段使用。

Yin 对案例研究的特点做了一些具体的补充［180］：

- 处理一些技术上的特殊情况，即变量远多于数据点，且得到一个结果。
- 依赖于多证据源，并需将数据汇集为三角剖分形式（Triangulating Fashion）或其他形式。
- 利用之前已得到认可的理论命题来指导数据的收集和分析。

因此，案例研究永远不会提供具有统计显著性的结论。相反是由各种不同种类的证据、图表、报表、文档连接在一起，支撑着一个强大而具有相关性的结论。

总而言之，案例研究的主要特点有［146］：

（1）它是一种柔性的研究方法，用于处理真实世界中具有复杂、动态特点的现象，如软件工程；

（2）其结论基于清晰的证据链，无论是定性还是定量的，都是以规划好且一致的方式从多个证据源获得；

（3）基于已建立的理论补充或建立新的理论补充已有知识。

5.1.1　为何要在软件工程中使用案例研究

软件工程领域包括软件及其相关制品的开发、运行和维护。软件工程研究很大程度上是在调研软件工程师和不同背景下的其他利益相关者是如何来进行开发、运行和维护工作的。无论是个人、团队还是组织进行软件开发，社会和政治问题对其都非常重要。也就是说，软件工程是一门交叉学科，涉及领域和案例研究一致，如心理学、社会学、政治学、社会工作、商业及社区计划（例如［180］）。这意味着软件工程的很多研究问题都适用于案例研究。

2.1 节对案例研究的定义强调了是对真实环境中的现象进行研究，尤其是在现象与其存在环境之间的边界不清晰时。而在软件工程中往往就是这样的。软件工程中的实验清楚地表明有许多因素都影响着软件工程活动的输出结果，例如 2.6 节中的重现
57 研究。案例研究提供了一种不需要严格划分被研究对象与它所处环境的研究方法；也许，理解的关键就在于这两者间的相互作用？

5.1.2　案例研究过程

进行案例研究有以下 5 个主要的过程步骤。

（1）案例研究设计：定义研究目标，制定研究计划。

（2）准备数据收集：定义数据收集的规程与协议。

（3）数据收集：在所研究的案例内进行数据收集。

（4）数据分析。

（5）撰写报告。

这个过程与几乎任意一种经验研究方法都基本相同，如针对实验方法，第 6 章的实验过程概述、第 7 ~ 11 章关于实验的详细阐述以及 Kitchenham 等［98］方法。不过，案例研究方法是一种柔性的设计策略，有大量的迭代步骤［2］。数据收集和分析都可以增量式进行。如果没有收集到足够的数据进行分析，就需要计划再次进行数据收集等。Eisenhardt 描述了从案例研究建立理论的过程，她在步骤 4 和步骤 5 之间加上了两步［52］：即形成假设和整理文献，其他概念与本文所述基本一致。

案例研究过程的 5 个步骤在 5.2 ~ 5.5 节详细陈述，其中数据收集的准备和收集过程在 5.3 节一起介绍。

5.2　设计和计划

案例研究是一种柔性研究方法，但这并不意味着就不需要计划了。相反，好的计划对案例研究的成功至关重要。有很多问题需要计划，如采用什么方法收集数据、访问组织的哪些部门、阅读哪些文档、访谈哪些人、访谈的频率，等等。这些计划可以在案例研究的协议中阐述，详见 5.2.2 节。

5.2.1 案例研究计划

一个案例研究的计划至少应该包含以下要素［144］。

- 目标：要实现什么？
- 案例：要研究什么？
- 理论：参考框架。
- 研究问题：需要知道什么？
- 方法：如何收集数据？
- 选择策略：到哪里寻找数据？

58

案例研究的目标可能是探索型、描述型、解释型或改进型的。目标自然更具通用性，没有确定性研究设计那样明确。目标最初更像是一个关注点，随着研究的进行而演化。研究问题阐述为了实现研究目的而需要知道的内容。与目标相似，研究问题也是在分析过程中不断发展的，它在分析过程中通过迭代最终收敛到具体的研究问题［2］。

在软件工程领域，案例可以是一个软件开发项目，这是最直接的选择。除此之外，案例也可能是一个独立的个体、一组人群、一个过程、一个产品、一条政策、组织中的一个角色、一个事件或者一项技术等。项目、个人或者小组也可以构成一个案例中的分析单元。类似于“玩具程序”的研究当然会因为缺乏真实的环境而被排除。

Yin［180］区分了整体性案例研究和嵌入式案例研究，前者将案例作为一个整体来研究，后者则对同一案例中的多个分析单元进行研究，详见图5-1。那么对于一项包括两个案例的研究，我们应该采取哪种方法呢？这取决于具体的研究情境和研究目标。比如，要研究的两个项目分别在两家不同的公司、两个不同的领域，都使用敏捷开发。这时，如果研究情境是普通软件公司，研究目标是研究敏捷方法的实践，那么该项目可以作为一个嵌入式案例研究中的两个分析单元。而如果研究情境是特定公司或特定应用领域，那么它们应被看作两个独立的整体性案例。

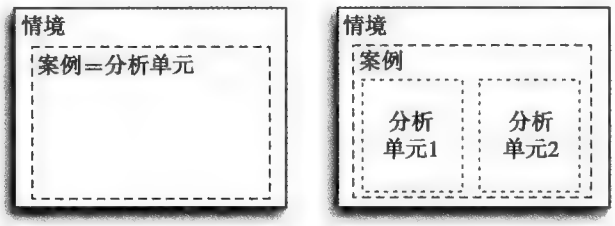


图5-1 整体性案例研究（左）和嵌入式案例研究（右）

在软件工程领域，依靠理论来发展研究方向的方式尚不成熟，详见2.7章。但是，定义研究的参考框架可以使案例研究的脉络清晰，有助于分析的进行及分析结果的评审。如果缺少理论支撑，也可以用分析中获取的观点和研究人员的背景知识作为参考框架。自然，基于理论的案例研究并不特指某种理论［38］。

59

案例研究中，尽管数据采集步骤的细节是在后期决定，但数据采集方法却是在设计阶段确定的。Lethbridge 等人［111］将数据采集方法分为三类：直接采集（如访谈）、间接采集（如工具）和独立采集（如文档分析）。具体细节将在 5.3 节进一步阐述。

在案例研究中，应有目的地选择案例和分析单元。这不同于调查和实验研究。调查和实验的主体是从总体中采样得到，目的是要保证研究结果的普适性。而在案例研究中，更期望案例的选择具有“典型性”“关键性”“启发性”，或者在某些方面是“独特的”［22］，并据此选择案例。对于比较性案例研究，必须选择在该分析所关注的属性上有变量的分析单元。但是实际执行中，案例研究的选择却和实验的选择［161］类似，许多案例是基于可获得性［22］进行选择的。

当要重复一个案例研究时，案例的选择尤为重要。所谓准确地重复一个案例研究，是指选择一个案例去预测相似的结果；否则就是理论上的重复，也即由于可预测的原因选择一个案例去预测有差异的结果［180］。

5.2.2　案例研究协议

案例研究协议包含案例研究中的设计决策，以及研究过程中的实施规程。该协议是一个持续更新的文档，每当案例研究计划变动时它都会随之更新。该协议服务于如下目的。

（1）指导数据收集，从而防止研究人员漏掉计划要收集的数据。

（2）制定协议的过程使研究在计划阶段具体化，有助于研究人员决定选用哪些数据源、问哪些问题。

（3）其他研究人员和相关人士可以通过阅读协议对计划进行反馈。举例来说，其他研究人员对协议的反馈可以降低由于以下问题带来的风险：漏掉相关数据源、漏掉访谈问题、漏掉研究所需的角色、忘记分析研究问题和访谈问题之间的关系等。

（4）可以作为一个日志，记录所有的数据采集和分析，以及由于研究本身的灵活性导致的决策的变化情况。这是后面撰写案例研究报告时重要的信息来源。为了跟踪研究项目中的各种变化，该协议应采用某种形式的版本控制。

表 5-1 总结了 Brereton 等人［25］提出的一个案例研究协议纲要。由此可以看出，
60 协议在细粒度上支持一个良结构的研究方法。

表 5-1　Brereton 等人［25］提出的案例研究协议纲要

部　分	内　容
背景	前期研究，主要和附加研究问题
设计	单个或多个案例，嵌入式或整体性设计；研究对象；从研究问题中提取出的命题
选择	案例选择的标准
规程和角色	现场过程；研究团队成员的角色
数据收集	识别数据、定义收集计划和数据存储

（续）

部　分	内　容
分析	解释说明、连接数据和研究问题、替代性解释所基于的准则
计划有效性	减小有效性威胁的策略
研究限制	指出还存在的有效性问题
报告	目标听众
进度安排	对主要步骤的估算
附录	所有细节信息

5.3 数据准备和数据收集

案例研究中使用的信息有几种不同的来源。为了减少因数据源单一而带来的信息误差，在案例研究中通过多种来源收集数据变得非常重要。如果同样的结论可以从多个信息源（如三角剖分法，将在6.2节介绍）得出，那么这个结论就比基于单一信息源的结论更可靠。在案例研究中，综合考虑不同角色的观点、探究诸如不同项目和产品案例之间的差别也是非常重要的。通常结论可以通过分析各数据源信息之间的差异得出。

根据 Lethbridge 等人的研究［111］，数据收集技术可以分为三个等级：

- 第一等级：直接法。在该方法中，研究者直接接触主体，并实时收集数据。访谈、观察组、德尔菲调查法［40］和应用“出声思考协议”［129］的观察法均属于这一等级。
- 第二等级：间接法。该方法也是直接收集原始数据，但在数据收集期间，研究者不与主体实际接触。通过软件工程相关工具记录日志和观察视频录像就是这一等级的方法。
- 第三等级：工作制品的独立分析。工作制品均已可用，有时使用编制成册的数据。通过分析文档（比如来自机构的需求规格说明书和故障报告）或机构数据库中的数据（比如计时数据）来收集数据的方法就属于这一等级。

第一等级方法的实施成本通常要高于第二等级或第三等级方法，因为它要求研究者和主体都要付出极大的努力。第一和第二等级的方法可以最大程度地控制收集哪些数据、如何收集、所收集数据的格式以及所收集数据的上下文，等等。第三等级方法一般花费较少，但是不能同等程度地控制数据收集，因此数据的质量不能得到有效控制，原始数据质量也不可控，并且这些数据对当前案例研究的可用性也存在不确定性。在许多情况下，研究者必须基于哪些数据可用，来决定数据收集的细节。对于第三等级方法，还应该注意到，由于数据是为了其他目的收集和记录的，而非当前研究的目的，因而不符合通用度量指南［172］。此外，采集数据时对数据有效性和完整性的需求是否和当前研究的一致也是不确定的。

在5.3.1～5.3.4节中，我们将讨论各种数据收集方法，包括访谈、观察、存档数

61

据和度量标准，这些方法都适用于软件工程案例研究［22，146，180］。

5.3.1　访谈

在基于访谈的数据收集过程中，研究者会对一组访谈对象（主体）提出一系列问题，这些问题都和该案例感兴趣的领域相关。大多数情况下，一次访谈针对一个单一主体，但也可以进行小组访谈。一系列访谈问题引导研究者和主体之间的谈话。

访谈问题是基于研究问题的（尽管表达方式不同）。问题可以是*开放性的*，即允许并邀请访谈对象在宽泛的范围内回答或者讨论；也可以是*封闭性的*，即提供一组答案让访谈对象选择。

访谈可以分成*非结构性的*、*半结构性的*和*全结构性的*三种类型［144］。对于非结构性访谈，访谈问题来自于研究者关心或者感兴趣的内容。这种情况下，访谈内容会基于访谈对象和研究者的兴趣意向进行。对于全结构性访谈，所有问题都会提前设计好并按照已设计的顺序进行。全结构性访谈在很多方面都与基于问卷的调查类似。对于半结构性访谈，虽然问题是已经设计好的，但提问顺序可以随意进行。访谈内容的进展情况可以灵活决定问题顺序，并且研究者可以把问题列表作为一个检查列表，来确保所有问题都已经被提问，即和检查列表类似。在案例研究中，半结构化访谈是最常用的一种，因为它允许访谈对象即兴发挥和探索。三种访谈方式在表 5-2 中进行了总结。

62

表 5-2　访谈类型总结

	非结构性访谈	半结构性访谈	全结构性访谈
访谈焦点	定性的回答	定性和定量的回答	寻找结构间的关系
访谈问题	在关注点的引导下进行	开放性和封闭性问题	封闭性问题
访谈目的	探索型	描述和解释型	描述与解释型

访谈过程可以分为几个阶段。首先，研究者提供访谈和案例研究的目的，并说明将如何使用从访谈中获得的数据。然后问一组引导式问题，如主体的背景情况等；相对来说这些问题都易于回答。引导完毕后，提出主要的访谈问题，这个过程占据了整个访谈的绝大部分。如果访谈涉及个人问题或者敏感的问题，比如对经济、对同事的看法，某些事情出现差错的原因，或者涉及被访谈者个人能力的问题等［80］，访谈者应该使被访谈者相信访谈的保密性，同时还要确保访谈者得到了被访谈者的信任。在建立信任之前，不建议访谈者提出上述相关问题。建议访谈者在访谈结束前对此次访谈的主要收获进行总结，这样既能获得反馈，也能避免不必要的误解。

在访谈过程中，建议采用音频或者视频的方式记录访谈内容。因为很多情况下，记笔记的方式不可能记录所有细节，而且在访谈过程中很难了解哪些内容是记录重点。访谈过程被记录下来后需要转化成文本用于分析。某些情况下，经过访谈对象审查过

的文本会更具价值。

在访谈研究的计划阶段就应确定访谈对象。鉴于案例研究的定性性质，在选择访谈对象时，建议不求同、但求异，如5.2节所述。访谈的多样性意味着访谈包括不同角色、不同性格的对象。访谈对象的数量可在研究过程中确定。确定访谈对象数量是否充分的一个标准是“饱和”，也就是说当加入新的访谈对象时，访谈不会得到更多的信息或观点[38]。

63

5.3.2 观察

为了调查软件工程师如何执行某个特定任务，我们还可以采用观察法。根据前面的分级标准，该方法属于第一或第二等级的方法。观察法包括很多具体方法，其中一种是用摄像机记录一组软件工程师的行动，随后再来分析这些录像；另一种方法是应用“出声思考”协议，在整个观察过程中，研究者重复地问问题（类似于“你有什么策略”“你正在思考什么”这样的问题）以提醒观察对象出声思考。这种方法可以与记录音频和击键的方法（由Wallace[176]等人支持的一种方法）一起使用。而会议中的观察法则是另一种类型，参与人员可以通过彼此交流来获取观察对象的信息。Karahasanović等人[93]还提出了另一种方法，使用特定的采样工具来收集参与者的数据和反馈。

根据研究者交互程度和被观察者感知被观察的程度，可以将观察的方法分为四类，详见表5-3。

表5-3 观察的不同方法

	被观察者感知被观察的程度高	被观察者感知被观察的程度低
研究者交互程度高	第1类	第2类
研究者交互程度低	第3类	第4类

第1或第2类观察法通常用在行动研究和经典民族学研究中。在这两类研究中，研究者也是团队成员，而不仅仅是其他团队成员眼里的研究者。第1类和第2类方法的不同之处在于：在第1类方法中，研究者是个“观察参与者”，然而在第2类方法中研究者则更多地被看作“普通参与者”。在第3类方法中，研究者只是个研究人员。一般说来，第3类观察方法使用第一等级的数据收集技术（比如上面介绍的“出声思考”协议）。而在第4类观察法中则更多地使用第二等级技术，比如视频记录法（有时又称为影像故事法）。

观察法的优点是：可以得到对研究现象的深层次理解。但这样的理解终究是由观察所得，也不免令人质疑“所见”与“真实”之间是否存在偏差[142]。另外不容忽视的是观察法会产生大量的数据，对这些数据进行分析需要消耗大量的时间。

64

5.3.3　归档数据

归档数据通常是指组织中的会议时间、不同开发阶段的文档、故障数据、组织结构图、财务记录，以及其他一些之前收集的度量数据。

归档数据属于案例研究中收集的第三等级的数据。配置管理工具是归档数据的重要来源，因为它能收集大量不同类型、不同版本的文档。对于其他的第三等级数据，需要注意的是这些文档在撰写时并没有计划作为研究数据使用。尽管研究人员可以通过调查最初收集这些文件的目的以及访问组织中的有关人员来获取一些信息，但是依然很难评估这些数据的质量。

5.3.4　度量标准

以上提到的数据收集技术大多关注定性数据。而定量数据在案例研究中也是很重要的。案例研究中，有时候需要从头定义数据并据此去收集数据，当然有时候也可以使用现有数据。显而易见的是，前者有更大的灵活性，并且这种量身定制的方式收集到的数据更能满足研究问题的需要。决定收集哪些数据应该依据面向目标的度量技术（比如目标问题度量法［11，172］，详见第3章）而定。

现有数据的例子有之前项目的工作数据、产品销售数据、以失效情况描述的产品质量度量数据等，这类数据也许可以在组织的度量标准数据库中获得。然而，需要注意的是，研究者既不能控制也不易评估数据的质量，因为最初收集这些数据是为了其他目的，当这些数据用于不同类型的分析时，极有可能缺失一些重要的信息。

5.4　数据分析

5.4.1　定量数据分析

针对定量和定性数据所采用的数据分析方法也是不同的。对于定量数据，分析方式通常包括描述性统计分析、相关性分析、建立预测模型和假设检验。所有这些方法都可用于案例研究。定量数据分析主要用在实验环境下，详见第10章。

65

描述性统计量，如平均值、标准差、直方图和散点图，可用于理解已收集的数据。相关性分析和建立预测模型是为了描述一个新过程活动的度量数据与之前过程度量之间的相关性。假设检验是为了确定一个或多个变量（独立变量）对一个或多个其他变量（非独立变量）是否有显著性影响。

应当注意的是，定量分析方法会提前假定一个固定的研究设计。例如，对于一个答案为定量数据的问题，如果当访谈进行到一半时问题被更改，那么将难以解释所收集到数据的平均值。此外，受调查者或测量点的数量限制，来自单个案例的定量数据集往往非常小，这需要在分析中特别关注。

5.4.2 定性数据分析

定性分析的基本目标是从数据中得出结论，并保持证据链清晰。证据链意味着读者能够理解从收集的数据推导出结果或结论的过程［180］。这意味着要充分地列出研究中的每个步骤和研究者做出的每个决定所需要的信息。

此外，定性研究分析具有两大特征：数据分析与数据收集过程同时进行；需要应用系统分析技术。由于方法的灵活性以及分析过程中常有新的发现，分析必须与数据收集同时进行。为了调查这些新发现，必须收集新的数据，并且一些基础工具，如访谈问卷，也要随之进行更新。由于数据收集技术频繁更新，而同时又要维护证据链，因此需要采用系统化分析技术。

为减少研究人员的个人偏见，分析最好由多名研究人员来完成。每名研究人员的初步结果汇总成共同分析结论。对合作计划的跟踪及报告有助于提高研究的有效性。

通用分析技术。定性数据的分析技术包括两类：假设生成技术和假设确认技术［148］。

假设生成技术用于从数据中发现假设。当使用这类技术时，研究人员应当摈除偏见，对数据中发现的任何假设保持开放态度。这些技术的结果同样也是假设。假设生成技术的例子有“常量比较”和“跨案例分析”［148］。假设确认技术是用来确认假设为真的技术，例如通过对更多数据进行分析。三角剖分和重复研究都是假设确认方法的例子［148］。负面案例分析试图找出拒绝假设的备选解释。这些基本技术类型往往迭代和组合使用。比如，首先生成假设，然后确认假设。假设可以在案例研究的一个循环中产生，或者结合来自分析单元的数据而产生，而假设确认则可能需要另一个循环或分析单元的数据来完成［2］。

66

这意味着定性数据分析包括一系列步骤（基于 Robson［144］）。首先，对数据进行编码，这意指为部分文本指定编码，这些编码能够表示特定的主题、区域、结构等。一个编码通常分配给许多文本片段，一个文本片段也可以分配多个编码。这些编码形成层次结构和子编码。编码后的材料可以与研究人员的评论和反馈（即“备忘录”）结合起来。完成这部分内容后，研究人员可以仔细检查这些材料从而获取第一组假设。假设可以是从材料的不同部分找到的相似短语、数据中的模式、主题下子群的差异，等等。正如上文所述，当实施进一步的数据收集时就可以使用这些假设，也就是数据收集和分析并行执行。在迭代过程中，制定一个小的归纳集，并最终形成形式化的知识体，这就是研究人员试图获取的最终结果。这显然不是一个简单的步骤序列。相反，它们是迭代执行并相互影响的。

表格是一种有用的数据分析技术，用表格组织编码后的数据，就可以得到一个数据概览。例如在用表格组织数据时，行代表兴趣编码，而列代表访谈主体。但如何组织数据则应根据每个案例的情况决定。

已经有一些专门的软件工具用来支持定性数据分析，如 NVivo[⊖] 和 Atlas[⊜]。但在一些情况下，处理文本数据采用诸如文字处理工具和数据表格工具等常用工具就可以了。

形式化水平。如前所述，在定性分析中结构化方法很重要。但数据分析的形式化水平可以不同。罗伯森［144］提及了下列方法。

- *沉浸方法*：最弱的结构化方法，其结构化水平非常低，更依赖于研究者的直觉和解释技巧。这些方法很难保持和表达证据链。
- *编辑方法*：该类方法包括很少的先验编码，即编码是基于研究人员在分析中的发现来定义的。
- *模板方法*：该类方法更形式化，也包含更多基于研究问题的先验知识。
- *准统计性方法*：该类方法非常形式化，并包含如词汇和短语等的频率计算等。

根据我们的经验，编辑方法和模板方法更适合软件工程案例研究。在非形式化的沉浸方法中，很难表示和获取一条清晰的证据链，且这种方法也很难表示某些结果，如文档和访谈中的词频分析。

5.4.3　有效性

研究的有效性是指研究结果的可信度，以及研究结果的适用范围和它不受研究人员主观看法影响的范围。当然，不能等到进入分析阶段才去考虑有效性。在案例研究的所有前期阶段都必须考虑有效性。

有许多不同的方法对文献的有效性和有效性威胁进行分类。这里我们选择一种分类模式，Yin［180］也曾经用这种方法做过案例研究，它类似于软件工程控制实验中经常采用的分类方法，详见 8.7 节。有一些研究者认为应该采用不同的分类模式去实现柔性的设计研究（可信性、可转移性、可靠性和可确认性），但我们更倾向于通过调整这个模式来实现柔性设计研究，而不再去修订术语表［144］。该模式区分四个方面的有效性，如下所述。

- *结构有效性*：反映研究中选择使用的度量可以真实反映研究者想法的程度，以及根据研究问题要去开展什么调查研究的程度。例如，对于访谈问题中的结构，如果研究人员与受访人员有不同的解释，那么就认为存在一个结构有效性威胁。
- *内部有效性*：主要用于考察因果关系时。当研究人员在分析一个调查因素是否会影响另一个调查因素时，很可能所调查的因素还会受到第三个因素的影响。如果研究者不知道存在第三个因素，或者不知道它影响调查因素的程度，那么就认为存在一个内部有效性威胁。
- *外部有效性*：主要关注研究结果可能的推广范围，以及本研究案例以外的人对该研究结果的感兴趣程度。在外部有效性分析中，研究人员要尝试去分析研究

⊖ http://www.qsrinternational.com。
⊜ http://www.atlasti.com。

结果与其他案例的相关程度。在案例研究中，不存在一个总体，可以从中提取出具有统计代表性的样本。但是案例研究的目标还是希望能够将研究结果扩展到具有共性的案例中，然后，根据研究结果的相关性提出一些理论。 68

- 可靠性：主要关注数据和分析结果对研究人员的依赖程度。假设，将来有另一个研究人员也进行了相同的研究，那么他得到的结果应该与当前研究的结果完全相同。常见的可靠性威胁有，如何处理所收集的数据不清楚，或者调查问卷或访谈问题不清晰的问题。在定量分析中，与可靠性对应的是结论有效性，详见8.7节。

如前所述，一定要从一开始就考虑案例研究的有效性。提高有效性的方法有多种，比如三角剖分法；开发和维护一个详细的案例研究协议；对设计、协议等进行同行评审；将收集的数据和得到的结果提交案例主体审阅；投入足够的时间进行案例分析，同时要足够重视“反面案例”的分析，如寻找与结果相矛盾的理论等。

5.5 撰写报告

报告应该和经验研究相一致。报告既是对研究结果的展示，也是用于评判研究质量的主要信息源。报告可能有不同的受众，比如同行研究者、政策制定者、研究资助者和业界从业人员［180］。因此可能需要为不同的受众撰写不同的报告。这里主要关注以同行研究者为主要受众的报告，比如期刊或会议文章，还可能包括技术报告［22］。Runeson 等人［146］给出了以其他受众为主的、不同格式的软件工程案例研究报告撰写指南。考虑到案例研究所产生的大量数据，Benbasat 等人认为“书籍或专题论文是发布案例研究的更好途径”［22］。

由于高层结构更灵活且大多基于定性数据，而低层细节不够规范，且更依赖于个案，因此案例研究大多采用高层结构（详见第11章）。下面，我们先讨论案例研究报告的特点，然后讨论推荐的结构。

5.5.1 特点

Robson 定义了一组案例研究报告应具备的特点［144］，可归结为如下几点。

- 说明研究什么。
- 对被研究案例的清晰描述。 69
- 提供“调查历史”，以便读者能知晓做了什么、谁做的以及是如何做的。
- 按规定格式提供基础数据，以便让读者理解结论的合理性。
- 清晰地表达研究结论及其适用环境。

此外，研究者发表其研究成果还要权衡好职责和目标、公司和个人的信誉［3］。

案例研究的目的和研究问题通常以开门见山的方式报告。如果在研究过程中发生了实质性变化，则应在报告中说明以便帮助读者理解案例。

案例的描述可能会比较敏感，因为这可能会使案例或者其主体被识别出来。例如

“瑞典的一家大型电信公司”就非常有可能是指爱立信公司的子公司。然而，除仅由应用领域和国家来描述案例之外，还可用更好的方式来刻画案例。通常情况下，内部特征，如所研究组织的规模和成员的平均年龄，可能要比领域和营业额这样的外部特征更有意义。否则说一个案例包括一个大公司旗下的子公司，那会导致无法区分具体指哪个子公司，或者说是一个小公司，那从许多候选公司中辨别出此小公司也是相当困难的。因此，描述案例时一定要权衡。

本质上，提供“调查历史”比纯粹的方法报告（例如，“我们使用半结构化访谈方法进行了一个案例研究。”）有更多的细节要求。因为研究的有效性和“做了什么、谁做的和如何做的”密切相关，因此需要报告研究过程中动作序列和各角色的工作。另一方面，报告中不可能有足够的篇幅去详细地描述案例研究中的每一个细节，因此必须平衡好。大量数据是在定性研究中收集的，分析这种数据的主要重点是精简和整理数据，以提供结论的证据链。然而，为了在研究中建立信任，还需要把用以支持结论的相关数据的快照提供给读者。这些快照可以是引用（典型或特殊声明）、图片或匿名主体的叙述。而且，在数据分类中使用的类别可以帮助读者理解证据链。

最后，一定要报告结论以及其适合的上下文环境，例如通过形成理论的方式。一个案例研究不能作为一个总体的代表进行推广，但毕竟总体样本法也不是获取和转移知识的唯一方式。没有统计分析也可以得出结论，这些结论可以被解释并且可以与其
70 他案例相关联。在软件工程领域内，通过理论的方式来交流研究成果还是一个有待开发的方法［72］（2.7 节已讨论过）。

5.5.2　结构

针对案例研究的学术报告，线性分析结构（问题、相关工作、方法、分析和结论）是最易被接受的结构。Jedlitschka 和 Pfahl 提出的用于软件工程实验的高级结构报告方法［86］也可以用于案例研究。然而，根据案例研究的具体特征和基于评估（指 Kitchenham 等人倡导的评估方法［101］）得出的其他问题，该方法还需要进行一些改动，最终的报告结构见表 5-4。

表 5-4　用于案例研究的推荐报告结构（基于 Jedlitschka、Pfahl 和 Runeson 等人［146］的研究）

章节标题	次级标题
题目	
作者	
结构化摘要	
引言	问题陈述 研究目的 上下文情境
相关工作	早期研究 理论

（续）

章节标题	次级标题
案例研究设计	研究问题 案例与主体的选择 数据收集规程 分析规程 验证规程
结果	案例和主体的描述、覆盖执行、分析和解释 分段，可以结构化。比如根据编码模式，把观察情况和相应的结论进行关联。 有效性评价
结论和展望	结论概述 与已有论据之间的关系 影响与贡献 限制 将来的工作
致谢	
参考文献	
附录	

在案例研究中，理论可以构成一个分析框架；因此，有两种相关工作：当前主题的早期研究和当前研究基于的理论。设计部分相当于案例研究协议，即，其中的案例研究计划包括为确保研究的有效性而采取的措施。由于案例研究是柔性设计的，而数据收集和分析又是错综复杂的，因此这些内容可以综合为一个小节（像5.3节中的那样）。

因此，低级结构的内容必须要调整为表5-4推荐的那样。尤其对于合并后的数据部分，编码方案经常包含自然的分段结构。或者对于一个具有比较性的案例研究，数据部分可以根据对照案例进行结构化。而对于纵向研究，数据结构可以按照时间尺度来组成。这种组合的结果部分还包括一个对最终结果有效性的评价。

下一章将会概述实验的实施过程，而每一步的详细过程将在后续章节介绍。

5.6 练习

5.1 案例研究在什么情况下使用？

5.2 作为一个柔性研究方法，计划在案例研究中起什么作用？

5.3 一项研究的案例选择标准是什么？

5.4 列举三种访谈形式，并说明使用场景。

5.5 描述定性分析的一般过程。

第6章
Experimentation in Software Engineering

实验过程

实验不是一件简单的事情。我们必须要合理地准备、实施和分析实验。实验最主要的优点是可以对主体、客观对象以及实验设备进行控制，以确保我们有能力得到比较通用的结论。此外，我们还可以使用假设检验方法进行统计分析，并且有机会可以复现实验。为了确保发挥这些优点，需要有一个过程来支持我们正确地进行实验并达到目标（这里的实验包括准实验，除非另有明确说明），实验的基本原则如图 6-1 所示。

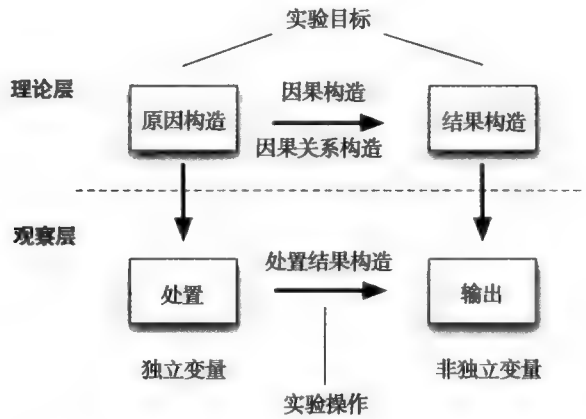


图 6-1　实验的原则（由 Trochim［171］改编）

实验的起点是我们假设存在一个因果关系，也就是说我们相信在构造的原因和结果之间存在某种关系，我们希望证明一个理论或者能够给出一个假设。这里假设是一种方法，可以形式化地表达某种设想，譬如关系。

实验可以用来验证我们的设想。例如，建立一个实验来检验理论或者假设，在设计时，构造若干可以控制的实验处置（被研究的变量可能的取值，见后文），完成处置并观察得到结果，这也意味着实验的目的是检验处置和结果之间的关系。如果实验构建正确，就能够推断出在假设中所描述的因果关系是否存在。

大部分实验的主要目的是评价某个假设或者关系，见 2.4.1 节。假设检验一般是针对前者，后者主要是基于所收集的数据建立一个关系模型，这个模型通常可以用多元统计方法得到，例如回归技术，然后用实验进行评价。本书主要关注假设检验，多元统计方法可以参见 Kachigan［90，91］和 Manly［118］等。

本章所介绍的实验过程凝练了前人的经验，期望以合适的活动确保实验成功。很多失败的案例都是由于我们在实验前忽视了某些因子，导致计划中遗漏了相应的分析而无法得到有效的结论。制定这个过程的目标就是为了支持实验的建立和实施。本章

简单概述实验的活动，具体细节请参见第7～11章。

6.1 变量、处置、对象和主体

在讨论实验过程之前，有必要先介绍一些定义，以建立实验的词汇表。当实施一个正式的实验时，我们会改变过程的输入变量，并研究由此而产生的结果。所以，实验中有两种变量，独立变量和非独立变量，如图6-2所示。

图6-2 独立变量与非独立变量图示

随独立变量变化而变化的变量，称为非独立变量（dependent variable，也叫响应变量，response variable），通常在一个实验中只有一个非独立变量。所有在过程中可以控制和操纵的变量，都称为独立变量（independent variable）。

示例：我们想研究新的开发方法对个人生产率的影响。判断是否可引入面向对象的设计方法替代面向功能的方法。实验中的非独立变量是生产率，独立变量可以是开发方法、参加实验的人员、支持工具和环境等。

实验的目的是研究改变一个或者多个独立变量时，对实验结果的影响，这些变量称为因子（factor）。实验中，需要将其他的独立变量控制在一个固定的水平，否则我们无法说明是哪个变量导致了结果。一次处置（treatment）为一个因子产生一个特定的值。

示例：在上面的实验案例中，因子是开发方法，因为我们希望研究改变开发方法的效果，我们对这个因子用两个处置，分别采用旧和新的开发方法。

实验设计（experiment design）时要考虑如何选择处置，以及其他独立变量应该设置的水平，见图6-3。实验设计将在第8章详细介绍。

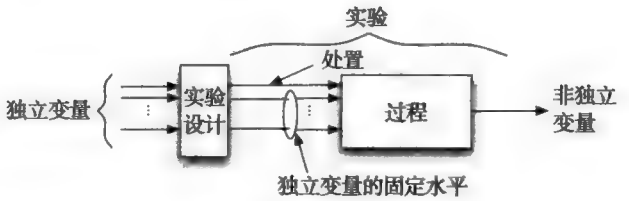


图6-3 实验图示

处置应用于对象（object）和主体（subject）的组合，例如，对象可以是一个将被不同审查技术评审的文档，采用这个处置的人便是主体[⊖]。在实验中对象和主体都是

74
↓
75

⊖ 有时会用术语“参与者”代替术语“主体”。当用“主体”时，主要表示人要采用不同的处置并进行相关的分析。当用“参与者”时，主要表示如何鼓励和促进人员参与实验。

独立变量。

示例：上述实验中，对象是要开发的程序，主体是程序员个人。

实验中包含一组检验（test）（有时也称作试验），每个检验都是一个处置、主体和对象的组合。需要注意的是，这里的检验不要和统计检验相混淆，我们会在第 10 章进一步讨论。检验的次数会影响实验的误差，所以通常会估计每个实验因子的均值。实验误差的大小也可以帮助我们树立对实验结果的信心。

示例：一个检验可以是 N 个人（主体）使用新开发方法（处置）开发程序 A（对象）。

在面向人的实验中，人是主体，使用不同的处置作用于对象。面向人的控制实验会有一些特殊性。首先，人有不同的技能和能力，本身就可以是独立变量。其次，人会不断学习，这也就意味着如果主体采用两种方法，其采用顺序会对结果产生影响，而且两次处置中的对象也不可能是完全相同的。第三，由于人具有猜测实验期望的能力，并有不同的主观动机，等等，会对以人为主体的实验带来各种人为的影响和威胁。因此如何选择和处理主体对实验结果非常重要。

面向技术的实验则比较容易控制，因为技术是确定的。在这种类型的实验中，不好控制的独立变量可以用选择不同的对象来替代。譬如一种工具或技术，可能只适用于某类程序。因此如何选择对象对实验结果就非常重要。

6.2　过程

过程提供了执行活动的步骤，譬如软件开发。过程之所以重要，是因为可以把过程看作是做什么以及如何做的检查单和指南。要完成一个实验，必须执行一些步骤，并且必须按一定的顺序执行。所以，我们需要一个过程，定义开展实验需要执行的活动。

这里介绍的过程主要针对实验，不过，如 5.1.2 节介绍的案例研究过程一样，同样的基本步骤在经验研究中也是需要的，主要的区别在于活动中具体的工作。譬如，调查法、实验和案例研究的设计不同，但它们都需要设计。案例研究的设计灵活，可以多次迭代地执行过程步骤，而调查法和实验的设计则比较固定，通常只执行一次过
76 程步骤。而且，这些基本的过程也可以用于其他的研究类型，只是应该做适当的裁剪以适应具体的研究，例如使用电子邮件的调查、大型软件开发的案例研究等。这个过程也适用于随机实验和准实验。后者在软件工程中经常用到，特别是当对主体（参与者）随机样本不可行的时候。

实验的起点是认识和理解，提出一种思路来进行实验去验证我们感兴趣的事情。换句话说，我们必须认清进行的实验对于我们正在调研的问题是合适的。这并非总是显而易见的，特别是，经验研究方法还没有频繁地应用于计算机科学和软件工程领域[170，181]。有一些争议讨论为什么计算机科学家应该更多地做实验，见 Tichy[169]。如果假设我们已经认识到实验是合适的，那么非常重要的一点是，必须仔细

地计划实验，以避免一些不确定性导致误解，见2.9节。

实验过程可以划分为以下主要活动，第一步是确定范围，我们需要根据问题、目标和目的确定实验的范围。然后是计划，需要设计实验，考虑实验需要的设施、工具和环境，并且评估实验的威胁。设计之后是操作，在操作活动中，要收集度量。然后在分析与解释活动中进行分析与评估。最后，在归档与展示活动中对结果进行归档与展示。这些活动如图6-4所示，下面会进一步解释并在第7~11章详细讨论。图6-5概要给出了实验过程所包含的活动。

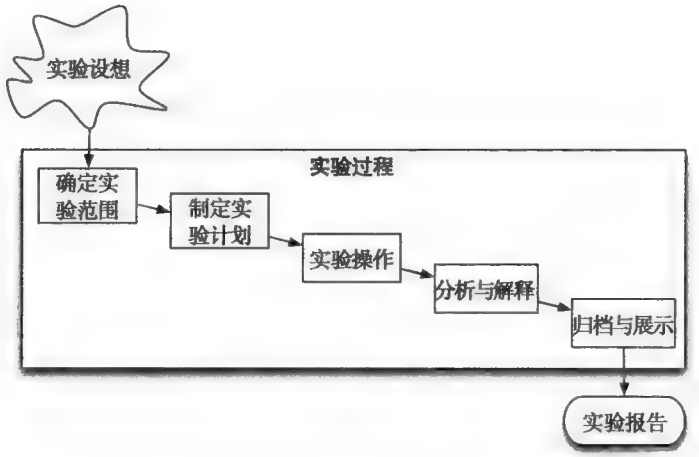


图6-4 实验过程概览

这个过程并不是一个“真的”瀑布模型，它并不假定一个活动必须在下一个活动开始前结束。过程中活动的顺序主要是这些活动开始的顺序。也就是说，这些活动可以部分迭代进行，但可能需要在继续实验之前返回并改进前面的活动。这其中的例外是一旦操作开始，就不可能再返回到确定范围和计划的活动。这是因为开始操作就意味着主体已经被实验影响，如果返回计划后再进行操作，实验就可能不能再使用同样的主体。

确定范围（Scoping）。第一个活动是确定范围，要清楚地陈述假设，并定义实验的目的和目标。在这个阶段，并不需要形式化地描述假设，但必须清楚。实验的目的应来自要解决的问题。[13]建议了一个框架，帮助我们捕捉到合适的范围。该框架包括下列成分。

- 研究对象（研究什么?）
- 目的（意图是什么?）
- 质量焦点（研究要达到的效果是什么?）
- 视角（从什么角度看?）
- 情境（实验场所是什么?）

第7章将进一步讨论这些问题。

计划（Planning）。计划活动是实验的基础。要详细确定实验的场景，包括人员和

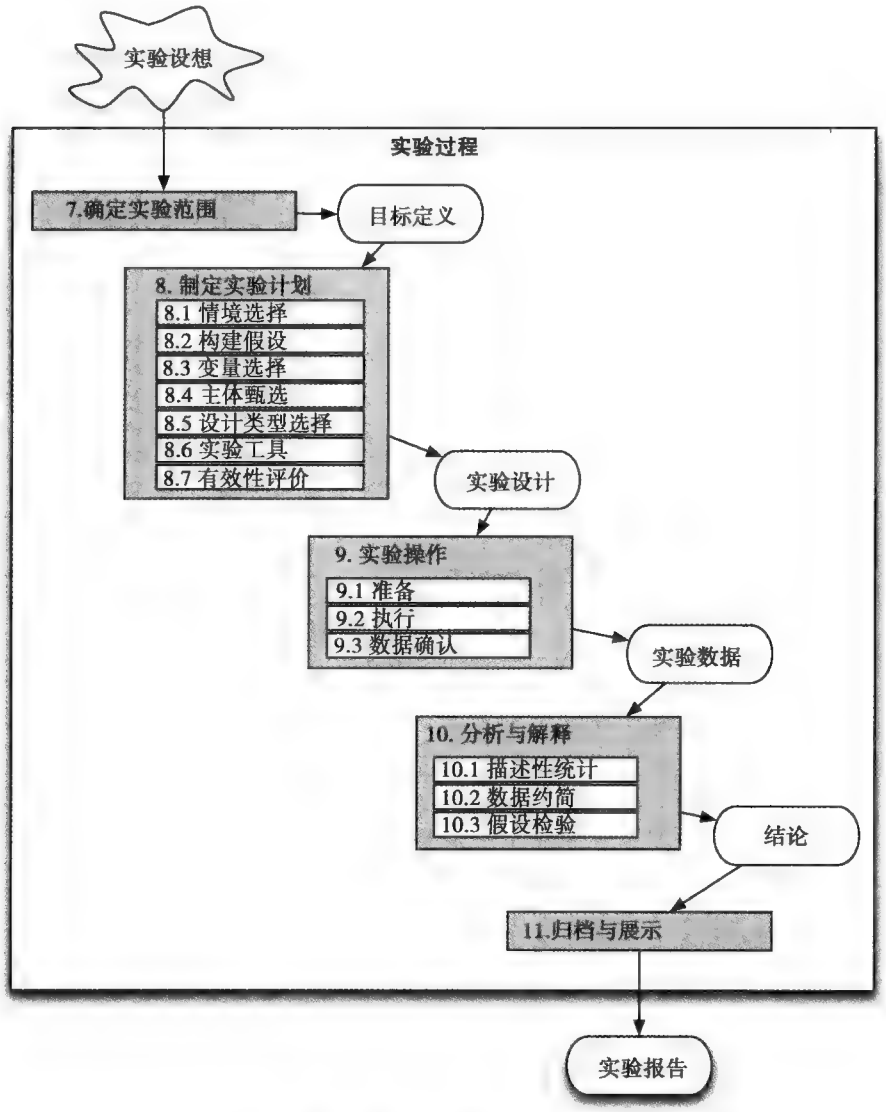


图 6-5　实验过程与本书章节相关制品的概览

环境，譬如是有学生参与的大学环境，还是工业环境。此外，实验的假设要形式化地表述，包括原假设和备择假设。

计划活动中的第三步是确定变量，包括独立变量（输入）和非独立变量（输出），这里重要的事情是要确定变量的实际取值，以及确定合适的度量尺度，这可以让我们在度量方法上设置一些约束，以便支持后面的统计分析。研究的主体也要在这里识别和确定。

下一步是实验设计，包括选择合适的设计方法，如主体的随机化。与设计密切相关的事情包括准备实验的设备、工具等。我们必须准备合适的实验对象，必要时还要开发指南，并定义度量规程，这些问题将在第 8 章具体讨论。

作为计划的一部分，还必须考虑如何保证我们期望的结果是有效的。有效性可以

分为四种主要的类别，即：内部、外部、构造和结论有效性。内部有效性关心给定的环境和结果的可靠性；外部有效性关心研究结果的通用性。许多时候，我们希望表达实验的结果在实验所运行的环境之外也是有效的；结构有效性的关键是判断相关的处置是否合适地反映了原因构造，并且结果也真实地描绘了效果构造，见图6-1；结论有效性则主要关心实验处置和结果之间的关系，我们必须判断处置和结果之间是否存在某种关系。

77
|
79

计划是实验至关重要的一步，要确保由此而展开的实验是有价值的。糟糕的计划可能会毁掉一个创意良好的研究。

操作（Operation）。操作由三个步骤组成，分别是：准备、执行和数据确认。在准备阶段，我们要准备主体以及实验需要的材料，譬如数据收集表。要告诉参与者实验的意图，获得他们的许可和承诺。执行通常不是问题，重要的是要保证实验包括数据收集按计划和设计执行。最后，我们还要确保实际收集的数据是正确的，并有效地刻画了实验。操作活动的细节将在第9章讨论。

分析与解释（Analysis and interpretation）。执行活动收集的数据为这个活动提供了输入，在这里要分析和解释这些数据。首先应该通过描述性统计技术去理解数据。这些技术可以为数据提供一些直观可见的视图，帮助我们非正式地理解和解释获得的数据。

下一步要考虑是否应该约简数据集，譬如移除数据点，或者当一些变量提供了同样的信息时，减少变量的数目。有一些成熟的方法可以用来进行数据约简。

数据约简后，就可以进行假设检验了。可以根据度量尺度、输入数据的值、结果的类型来选择实际的检验方法。统计检验以及详细的解释性统计和数据约简技术在第10章讨论。

本活动的一个重要工作就是解释，也就是说，我们必须根据分析决定是否有可能拒绝设定的假设。这是我们决定如何使用实验结果的基础，并激发下一步研究的思路，例如是否应该进行更大范围的实验或者案例研究。

归档与展示（Presentation and Package）。最后一个活动关心的是归档和展示发现的结果。主要包括结果的撰写，可以是可公开发表的研究论文，可供重复实验的实验数据包，也可以是公司经验库的一部分。在这个最后的活动中，确保以适当的方式进行经验总结是非常重要的。此外，一个实验永远不会是一个问题的最终答案，能够重复这个实验非常重要，那么其前提就是必须有一个完整清楚的文档。不过，在使用实验数据包时还是应该非常小心，因为使用同样的实验设计和文档可能会引入原实验一些系统性的问题和偏差，如2.6节所述。总而言之，实验后我们必须花一些时间来撰写文档并以合适的方式展示。实验的展示将在第11章详细介绍。

80

6.3 总览

实验过程的这些步骤都会在后续的章节中详细讨论。第12章提供一个例子帮助我

们更好地理解这个过程。该例子将严格遵循这个定义的过程，以指导使用。实验过程的概要框架见图6-5。

6.4　练习

6.1　什么是因果关系？

6.2　什么是处置？为什么有些时候在一些随机序列中必须应用处置？

6.3　什么是独立和非独立变量？

6.4　什么是准实验？解释一下为什么在软件工程中准实验很常见。

81　6.5　实验过程有哪些主要步骤？为什么区分这些步骤很重要？

第二部分

Experimentation in Software Engineering

实验过程的步骤

第7章
Experimentation in Software Engineering

确定范围

实验是一个劳动密集型的工作，为了提高效率，确保实验的意图可以贯穿整个实验的始终非常重要。确定范围是建立实验的基础，如图 7-1 所示。如果基础不合适，就难免返工，甚至南辕北辙。确定范围阶段的目的就是按照一个设定好的框架定义出实验的目标，这里我们遵照 Basili 和 Rombach 提出的 GQM 模板［13］来定义目标。

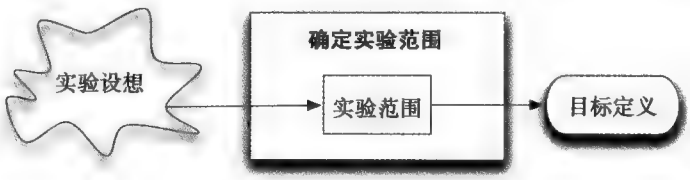


图 7-1　确定实验范围阶段概览

7.1 节讨论如何确定实验范围，7.2 节给出了示例。

7.1　确定实验范围

实验范围取决于实验的目标。目标定义模板的用处在于确保重要的实验元素在计划和开展实验之前都已考虑到并定义好。按照这样的模板定义目标，可以为实验建立正确且适当的基础。［13］提出的目标模板如下：

分析 <研究对象(Object(s) of study)> 中定义的对象

实现 <目的(Purpose)> 中要求的目的

具有 <质量焦点(Quality Focus)> 中期望的效果

从 <视角(Perspective)> 规定的角度关注这些质量焦点

在 <情境(Context)> 陈述的情境中进行实验

研究对象是实验中研究的实体，可以是产品、过程、资源、模型、度量或者理论，例如最终产品、开发或者审查过程、可靠性增长模型等。目的定义了实验的意图，例如评估两种不同技术的影响、刻画组织的学习曲线等。质量焦点是实验关注的主要效果，例如效益、成本、可靠性等。视角指从什么角度解释实验结果，例如开发者、项目经理、客户和研究者等。情境是实验运行的环境。情境应简要地定义哪些人（主体）参与实验以及会用到哪些软件制品（对象[⊖]）。主体的属性包括经验、团队规模、工作量等，对象的属性包括规模、复杂度、优先级、应用领域等。

⊖ 注意，此处的“对象”与之前定义的“研究对象”并不相同。

实验情境可以根据主体、对象的数量来进行分类［10］，见表7-1。

表7-1 实验情境分类

		对象数	
		一个	多个
每个对象的主体数	一个	单一对象研究	多对象变异研究
	多个	单对象多检验研究	主体－对象组合研究

单一对象研究（Single object study）在一个主体和一个对象中进行，多对象变异研究（Multi-object variation study）在一个主体作用于多个对象的情境下进行，单对象多检验研究（Multi-test within object study）的特点是多主体作用于一个对象，主体－对象组合研究（Blocked subject-object study）的情境则是一组主体和一组对象组合作用的情况。这些实验类型都可应用于实验或者准实验。准实验中主体或对象的选择一般缺乏随机化，譬如在单一对象研究中，如果主体和对象的选择不是随机的，则是一个准实验，否则就是一个实验。实验和准实验的区别在Robson［144］中有详细的讨论。

NASA-SEL［10］在净室原则和技术方面的一系列实验给出了不同实验类型的例子。净室集合了一组软件工程方法和技术，以达到生产高质量软件的目的。净室的简要介绍见Linger［112］，它是由四步组成的一个实验系列。第一步，以主体－对象组合方式进行阅读和单元测试实验［12］，见表7-2中的1；第二步，在学生环境中采用净室技术开发一个项目［149］，实验是单对象多检验的类型，见表7-2中的2；第三步，在NASA-SEL采用净室技术开发一个项目［14］，作为单一对象研究实验，见表7-2中的3；第四步，在同样环境中开展三个净室项目，组成一个多对象变异研究的实验［14］，见表7-2中的4。分析另外一个新技术时，从新的阅读实验开始下一轮实验［18］，见表7-2中的5。这个实验系列Linkman和Rombach也讨论过［113］。

86

表7-2 情境分类实验的示例，Basili［10］

		对象数	
		一个	多个
每个对象的主体数	一个	3. SEL净室项目1［14］	4. SEL净室项目2－4［14］
	多个	2. 马里兰大学净室实验［149］	5. 基于场景的阅读对比检查单［18］

表7-2的例子解释了如何在案例研究（见3和4）之前进行一些实验（见1和2）作为预研究。这符合2.9节和2.10节介绍的风险和成本平衡原则下技术的逐步转移和提升。

7.2 实验案例

根据不同的研究对象、目标填写目标定义框架，表7-3给出了一些元素的例子。

表 7-3　目标定义框架

研究对象	目的	质量焦点	视点	情境
产品	刻画	效率	开发者	主体
过程	监督	成本	修改者	对象
模型	评价	可靠性	维护者	
度量	预测	可维护性	项目经理	
理论	控制	可移植性	合作经理	
	改变		客户	
			用户	
			研究者	

通过组合框架中的元素可以构造一个实验，下面给出了一个案例。该案例定义了一个审查实验，目标是评价不同审查技术，如基于视角的阅读审查和基于检查单的阅读审查。基于视角的阅读审查由 Basili 等［18］提出，已经在多个实验中被评价过，如基于视角的阅读审查和 Maldonado 等［117］提出的已在 NASA 使用的阅读审查的对比比较，以及 Laitenberger 等［107］进行的基于视角和基于检查单的阅读审查方法的比较。研究者还比较了一些其他的阅读审查技术，例如 Thelin 等进行的基于用途和基于检查单的阅读方法比较［168］。

87

本案例的研究对象是基于视角的阅读（Perspective-Based Reading，PBR）技术和基于检查单（Checklist-Based Reading，CBR）的阅读技术。目的是评价这些阅读技术，特别是要考虑 PBR 中视角间的不同。质量焦点关注这些阅读技术的效果和效率，实验的视角是从研究者的角度，实验情境是以硕士和博士为主体，实验包是一组文本化的需求文档。因为涉及多个主体和需求文档，所以实验以主体 - 对象组合方式进行，见表 7-1。

实验案例总结如下：

分析 PBR 和 CBR 技术

目的是评价两个方法

关注其效果和效率

从研究人员的角度解释

情境为硕士和博士学生阅读需求文档

在第 8 ~ 10 章将继续用这个例子解释实验过程的其他步骤。这个总结构成了实验的目标定义，将作为实验过程中计划步骤的输入。

7.3　练习

7.1　为什么在实验的开始建立清晰的目标是重要的？

7.2　根据你想进行的实验，试举一个目标定义的例子。

7.3　实验情境为什么重要？

88

7.4　如何刻画实验情境？

7.5　试解释一个研究系列如何用作技术转移？

第8章
Experimentation in Software Engineering

计　划

确定实验范围后，就要着手制定实验计划。确定范围是实验的基础——明确为什么要做这个实验，而计划则是为如何开展实验做准备的。

与所有的工程活动一样，实验也要做计划，然后要按计划执行并控制实验，否则实验的结果会受到干扰甚至破坏。

实验的计划阶段可以分为7个步骤，其输入是已经定义的目标，见第7章。根据目标定义，在情境选择步骤中挑选开展实验的环境。接下来进行假设构建，然后在变量选择中确定独立变量和非独立变量，并进行主体甄选。根据假设和变量选择确定合适的实验设计类型。之后要准备合适的实验工具来实际开展实验，最后是实验的有效性评价。计划的过程可以是迭代的，直到完成实验设计。计划阶段的概览见图8-1。

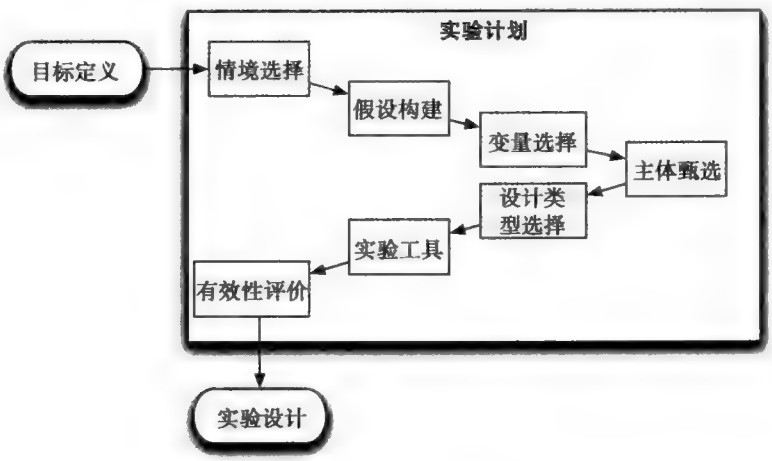


图8-1　计划阶段概览

8.1　情境选择

为了使实验结果具有最好的通用性，应该由专业人士在规模较大且真实的软件项目中开展实验。然而，实验总是有风险的，例如，新方法可能并不如愿，还会导致项目延期。通常可以在实际项目之外并行运行一个离线项目作为备选，这可以降低风险，但增加了额外的成本。一种经济的方式是用学生来执行这样的项目。这种项目既便宜也更容易控制，但较之由更多不同经验的专业人士执行的项目，前者更直接地设定了一个确定性的情境。此外，这样的项目很少解决实际问题，由于成本和时间的限制，其规模往往像个玩具。情境选择中的取舍取决于我们希望研究在特定的环境下有效，

还是在通用的软件工程领域内有效，进一步分析见 8.7 节。一些文献也讨论了用学生做主体开展实验的取舍和折中问题，如 Höst 等人的［77］。

因此，实验情境可以用以下四个维度来刻画：

- 离线与在线；
- 学生与专业人士；
- 玩具与实际问题；
- 特殊与通用。

一种常见的实验是将现有的某些东西和新的东西进行比较，例如，现有的审查方法和一个新方法的比较［18，136，139］。这种类型的研究通常与两个问题相关。第一，什么是现有的方法？它通常已经应用了一段时间，但没有很好地文档化，并且往往在应用中存在不一致问题。第二，学习新方法可能会影响现有方法的应用方式。

90

与之相关地，为了使结果有效，在计划时，我们还要考虑的问题是实验的参与者。

8.2　假设构建

实验统计分析的基础是假设检验。要形式化地陈述假设，并根据实验过程中收集的数据在可能的情况下拒绝假设。如果假设可以被拒绝，就可依据给定风险下的假设检验做出结论。

在计划阶段，实验定义可以进一步形式化地描述为假设，包括以下两个假设：

（1）原假设（Null）。原假设记为 H_0，表示在实验设定下，没有真正潜在的趋势或者模式，我们所观察到的不同是偶然的。这是实验希望以尽可能显著的统计意义拒绝的假设。例如，假设一种新审查技术发现故障的平均数目和以前技术发现的一样多，亦即 H_0：$\mu_{N\,\text{old}} = \mu_{N\,\text{new}}$，其中 μ 表示平均值，N 表示发现的故障数。

（2）备择假设（Alternative）。备择假设记为 H_a，H_1 等，是一个希望原假设不成立的假设。例如，假设新审查技术发现故障的平均数目大于以前技术发现的。亦即 H_1：$\mu_{N\,\text{old}} < \mu_{N\,\text{new}}$。

各种文献中介绍了大量不同的统计检验方法，可以用来评价实验的结果。应该根据上述假设的情况选择合适的统计检验技术，参见 10.3 节的详细介绍。

假设检验也有各种风险，可能拒绝了一个真假设，也可能接受了一个错误假设。通常称为 I 类错误和 II 类错误。

（1）I 类错误（Type-I-error）。I 类错误指统计检验认为存在的模式或者关系实际上是不存在的。亦即 I 类错误的概率可以表示为

$$P(\text{I 类错误}) = P(\text{拒绝 } H_0 \mid H_0 \text{为真})$$

在上面的例子中，I 类错误是指尽管两个方法发现故障的平均数目是一样的，但拒绝 H_0 的概率。

（2）II 类错误（Type-II-error）。II 类错误指统计检验认为不存在的模式或者关系实际上是存在的。亦即 II 类错误的概率可以表示为

$$P(\text{II 类错误}) = P(\text{不拒绝 } H_0 \mid H_0 \text{ 为假})$$

在上面的例子中，II 类错误是指尽管两个方法发现故障的平均数目不一样，但没有拒绝 H_0 的概率。 91

错误的大小取决于各种因子，譬如统计检验在收集的数据中揭示一个存在模式的能力，称为统计检验效能（power）。

统计检验效能是当 H_0 为假时，检验认为存在这个模式的概率。实验应该选择检验效能尽可能高的检验方法，效能可以表示为：

$$\text{Power} = P(\text{拒绝 } H_0 \mid H_0 \text{为假}) = 1 - P(\text{II 类错误})$$

所有因子都应该在实验计划时考虑到。

8.3 变量选择

在设计开始前，还需要选择独立变量和非独立变量。

*独立变量*指那些在实验中可以控制和改变的变量，正确地选择变量并非易事，通常需要专业知识。这些变量应该对非独立变量有影响，并且是可控制的。独立变量和非独立变量的选择通常是同时或者逆向进行的。独立变量的选择还包括度量单位的选择、变量的范围，以及检验的可信水平。

处置的效果用*非独立变量*度量，通常实验只有一个非独立变量，并且直接从假设中派生出来。有些变量通常不能直接度量，而必须用间接度量代替。要仔细验证间接度量，它们会影响实验结果。选择好独立变量后，可以进一步完善假设。独立变量的选择也意味着确定了度量单位和变量范围。只选择一个非独立变量的原因是，如果有多个非独立变量，则会出现“钓鱼与出错率”的现象，这会导致在讨论有效性威胁时，因情况太复杂而导致风险。具体见 8.8.1 节的介绍。

8.4 主体甄选

对开展实验的主体进行甄选是非常重要的［144］，将直接关系到实验的结果。要让实验结果在预先期望的范围内有效，选择的主体就必须要能代表这个范围，通常称为从某个总体中选择样本。

从总体抽样可以是概率抽样或者非概率抽样，两者的不同在于，选择每种主体的概率在概率抽样中是已知的，而在非概率抽样中未知。*概率抽样技术*（probability sampling technique）的例子如下： 92

- *简单随机抽样*（Random sampling）：从总体列表中随机选择主体。
- *系统抽样*（System sampling）：随机从总体列表中选择第一个主体，然后每隔 n 个选择一个作为主体。
- *分层随机抽样*（Stratified random sampling）：根据一个已知的分布，将总体划分为若干组或者层，在每层随机选择。

非概率抽样技术（non-probability sampling technique）的例子如下：

- 便利抽样（Convenience sampling）：选择最接近和最方便的人员作为主体。
- 配额抽样（Quota sampling）：从总体的不同元素中选取主体，而在每一种元素中一般用便利抽样进行选择。

样本规模对结果的影响也是很重要的，样本越大，结果出错的可能性越低。样本规模也同时会直接影响统计检验的效能，见 10.3.1 节。选择样本规模的基本原则如下：

- 如果总体的变化性较大，则需要的样本规模也较大。
- 数据分析可能影响样本规模的选择，因此也要考虑实验设计阶段的数据分析技术。

8.5　实验设计

应用统计分析方法分析收集到的数据并解释结果，才能得到有意义的实验结论。详细的介绍见第 10 章。要得到最好的实验，必须仔细地计划和设计实验。应用什么样的统计分析技术还取决于实验设计和使用的度量单位，见第 3 章。可见，设计和解释是密切相关的。

8.5.1　实验设计的选择

实验由一组对处置的检验组成。要想从实验中获得最大的收益，必须精心计划和设计一系列检验。实验设计中应该描述如何组织和执行这些检验。更正式地，我们可
93 以用一组检验来定义实验。

如上所述，设计和统计分析是密切相关的，设计的选择会影响分析，反之亦然。在设计实验时，要根据假设找到合适的统计分析技术，以拒绝原假设。要根据统计技术的适用条件（譬如度量的单位）以及应用的对象和使用的主体来进行实验设计。同时，要确保处置的效果是可见的。实验设计还要确定必须要进行的检验数量。合适的实验设计也是该实验可以重复的基础，以下两节，我们提出了一个通用的设计原则和一些标准的设计类型。

8.5.2　通用设计原则

实验设计时要考虑很多方面。通常的设计原则有随机策略、分块阻断和均衡设计，大多数的实验混合采用这些原则，下面我们用一个例子来解释这些常规的设计原则。

例子：某公司要做一个实验，调查若采用面向对象的设计代替公司标准的设计原则，对程序可靠性将产生的影响。用程序 A 作为实验对象，实验设计是“单对象多检验”类型，见第 7 章。

随机策略（Randomization）：随机策略是最重要的实验设计原则之一。所有数据分析的统计方法都要求观察到的数据来自于独立的随机变量。要满足这个需要，就必须

随机化。随机策略可应用于对象、主体的分配以及检验执行的顺序等。随机化是为了平均一些因素的影响，也可以用于选择代表总体利益的主体。

示例：从公司可用的设计人员中随机选取一些人员作为代表（这是实验的主体），并随机分配给每一个处置（面向对象设计和公司标准的设计原则）。

分块阻断（Blocking）：有些时候，某些因子对实验反应会产生影响，但我们对这种影响没有兴趣。如果这种影响是已知且可以控制的，就可以采用分块阻断的设计技术。分块阻断用于在处置比较时，系统性地消除一些不期望的影响，在同一块内，这些不期望的影响是相同的，所以我们可以研究同一块内不同处置的效果。分块阻断的目的是消除研究中不期望的影响，因此不关注块之间的影响。这种技术可以提高实验 94
的精度。

示例：参加实验的人员（主体）有不同的经验。一些人有面向对象设计的经验，而另外一些人没有。要最小化经验对实验结果的影响，这些人可以分为两组（两块），一组有面向对象设计的经验，另外一组没有。

均衡设计（Balancing）：如果给每个处置分配同样数目的主体，就是一个均衡设计。均衡设计是我们所期望的，因为它可以简化统计分析并增强分析的效能，但不是必需的。

例子：如果实验采取均衡设计，就意味着每个组（块）的实验人数是一样的。

8.5.3 标准设计类型

本节介绍最常见的实验设计。设计范围从只有单一因子的简单实验，到有多个因子的复杂实验。Montgomery 在［125］中深入讨论过实验设计，Juristo 和 Moreno 更进一步地在［88］中详尽阐述了软件工程的实验设计。针对大多数设计，本文给出一个示例简洁明了地阐述假设，并对每一个设计建议了适用的统计分析方法。本节介绍的设计类型适用于：

- 单因子双处置；
- 单因子多处置；
- 双因子双处置；
- 多因子双处置。

单因子双处置。这些实验希望对比两个处置，最常见的是比较每个处置非独立变量的均值，下面的标记经常用到：

μ_i 处置 i 的非独立变量的均值。

y_{ij} 处置 i 的非独立变量的第 j 个度量值。

实验示例：实验将调查是否一个新的设计方法生产的软件较之用以前设计方法生产的软件具有更高的质量。实验中的因子是设计方法，处置是采用新的和老的设计方法，非独立变量可以是开发中的缺陷数。

完全随机设计（Completely randomized design）。这是一个基本的实验设计，用于

比较两个处置的均值。设计时，两个处置作用于同样的对象，并且为每个处置随机指
95 派主体，见表 8-1。每个主体只使用一个处置产生对象。如果每个处置的主体数量相同，这个设计还是均衡的。

表 8-1　为随机设计给处置分配主体的例子

主体	处置 1	处置 2
1	X	
2		X
3		X
4	X	
5		X
6	X	

假设示例：

H_0：$\mu_1 = \mu_2$

H_1：$\mu_1 \neq \mu_2$，$\mu_1 < \mu_2$ 或者 $\mu_1 > \mu_2$

分析示例：t 检验，Mann-Whitney 检验，见 10.3 节。

成对比较设计（Paired comparison design）。有时，可以通过比较匹配的实验材料对来提高实验的精度。在这样的设计中，每一个主体采用两种处置产生同样的对象，有时也称为交叉设计。这种设计会面临一些挑战，我们将在 10.4 节中结合案例做进一步讨论。为了最大程度降低主体采用处置顺序而产生的影响，应该随机地给每个主体指定顺序，见表 8-2。这种方法并不一定适用于所有比较的例子，譬如当主体会从第一个处置获得太多执行第二个处置的信息时。实验的比较可以是观察成对度量的差异性是否为零。如果两个处置开始时的主体数量相同，这个设计也是均衡的。

表 8-2　为配对设计分配处置的例子

主体	处置 1	处置 2
1	2	1
2	1	2
3	2	1
4	2	1
5	1	2
6	1	2

假设示例：

$d_j = y_{1j} - y_{2j}$并且 μ_d是差异的均值。

H_0：$\mu_d = 0$

H_1：$\mu_d \neq 0$，$\mu_d < 0$ 或者 $\mu_d > 0$

分析示例：配对 t - 检验，符号检验，Wilcoxon 检验，见 10.3 节。

单因子多处置。和只有两个处置的实验一样，多处置实验也希望对处置结果进行

比较，通常是比较处置的均值。

实验示例：实验调查使用不同程序设计语言编写的软件的质量。实验因子是程序设计语言，处置可以是 C、C ++ 和 Java。 96

完全随机设计（Completely randomized design）。完全随机设计要求实验以一种随机的顺序进行，因此在实验环境中使用的处置要尽可能统一。设计要求所有处置作用于一个对象，而所有主体随机分配给各个处置，见表 8-3。

表 8-3 为主体分配处置的例子

主 体	处置 1	处置 2	处置 3
1		X	
2			X
3	X		
4	X		
5		X	
6			X

假设示例，这里 a 是处置数：

H_0：$\mu_1 = \mu_2 = \mu_3 = \cdots = \mu_a$

H_1：$\mu_i \neq \mu_j$，至少存在一个 (i, j) 对

分析示例：ANOVA（Analysis OF VAriance）方差分析检验与 Kruskal-Wallis 检验，见 10.3 节。

随机完全分块阻断设计（Randomized complete block design）。如果主体间的差异性很大，用随机完全分块阻断设计可以最小化其影响。在实验设计中，每个主体都会使用所有处置，因此形成了一个更加同构的实验单元。也就是说，在主体上对实验进行分块，见表 8 -4。块代表了随机化的结果。这种实验设计，让所有处置作用于一个对象，而主体使用处置的顺序是随机分配的，上面介绍的成对比较设计是当处置只有两个时的特例。随机完全分块阻断设计是最常用的一种实验设计。

表 8-4 为主体分配处置的例子

主体	处置 1	处置 2	处置 3
1	1	3	2
2	3	1	2
3	2	3	1
4	2	1	3
5	3	2	1
6	1	2	3

假设示例，这里 a 是处置数：

H_0：$\mu_1 = \mu_2 = \mu_3 = \cdots = \mu_a$

H_1：$\mu_i \neq \mu_j$，至少存在一个 (i, j) 对

分析示例：ANOVA 方差分析检验与 Kruskal-Wallis 检验，见 10.3 节。

双因子。当实验因子增加到两个时，实验会变得比较复杂。单因子实验中的单一假设会分解成三个。针对两个因子各有一个假设，另外一个针对两个因子之间的交互作用。通常使用下面的标记：

97

τ_i　　处置 i 在因子 A 的效果。

β_j　　处置 j 在因子 B 的效果。

$(\tau\beta)_{ij}$　　τ_i 和 β_j 交互作用的效果。

2×2 析因设计（2×2factorial design）。这种实验中有两个因子，每一个都有两个处置。设计时，将主体随机分配给每一个组合处置，见表 8-5。

表 8-5　2×2 析因设计的例子

		因子 A	
		处置 A1	处置 A2
因子 B	处置 B1	主体 4.6	主体 1.7
	处置 B2	主体 2.3	主体 5.8

实验示例：实验基于“好”和“差”的需求文档，分别调查用结构化和面向对象方法开发的设计文档的可理解性。第一个因子 A 是设计方法，第二个因子 B 是需求文档。因为有两个因子、两个处置，并且每种组合都是可能的，所以用 2×2 析因设计。

假设示例：

H_0：$\tau_1 = \tau_2 = 0$

H_1：至少存在一个 $\tau_i \neq 0$

H_0：$\beta_1 = \beta_2 = 0$

H_1：至少存在一个 $\beta_j \neq 0$

H_0：对所有 i，j，$(\tau\beta)_{ij} = 0$

H_1：至少存在一个 $(\tau\beta)_{ij} = 0$

分析示例：ANOVA 方差分析检验，见 10.3 节。

两阶段嵌套设计（Two-stage nested design）。对于一个因子（例如 A）的不同处置，如果另一个因子（例如 B）在实验中是相似但不完全相同的，则需要使用两阶段嵌套或者层次设计（Hierarchical design），因子 B 嵌套在因子 A 中。两阶段嵌套设计有两个因子，每个因子有两个或多个处置，实验设计和分析与 2×2 析因设计一样，见表 8-6。

表 8-6　B 嵌套于 A 的二阶段嵌套设计例子

因子 A			
处置 A1 因子 B		处置 A2 因子 B	
处置 B1′	处置 B2′	处置 B1″	处置 B2″
主体 1.3	主体 6.2	主体 7.8	主体 5.4

实验示例：实验调查单元测试对分别用函数式编程语言和面向对象编程语言编写的易错程序和不易错程序的测试效率。第一个因子 A 是编程语言，第二个因子 B 是程序的易错倾向。因为函数式编程语言的易错/不易错和面向对象编程语言的易错/不易错是不同的，所以这个实验的设计必须是嵌套的。

多因子。很多情况下，实验必须考虑多个因子，因此，非独立变量的效果不仅单独依赖于每个因子，还依赖于因子间的相互作用，这些相互作用可以是两个或多个因子的交互。这种设计类型称为析因设计。本节简要介绍这种设计类型，每个因子只有两个处置。各因子有多处置的设计见 Montgomery 的介绍［125］。

2^k析因设计（2^kfactorial design）。2×2 析因设计是 2^k 析因设计的特例，即 $k=2$ 的情况。2^k 析因设计有 k 个因子，每个因子有两个处置，也就意味着有 2^k 种不同处置的组合。必须检验所有的组合，才能评价 k 个因子的影响。主体应该随机地分配给各个组合，一个 2^3 析因设计的例子见表 8-7。

表 8-7 2^3 析因设计例子

因子 A	因子 B	因子 C	主体
A1	B1	C1	2，3
A2	B1	C1	1，13
A1	B2	C1	5，6
A2	B2	C1	10，16
A1	B1	C2	7，15
A2	B1	C2	8，11
A1	B2	C2	4，9
A2	B2	C2	12，14

这种设计类型的假设和分析与 2×2 析因设计一样，更多 2^k 析因设计的细节，Montgomery 在［125］中做了详尽的介绍。

2^k 部分析因设计（2^k fractional factorial design）。在 2^k 析因设计中，随因子数目的增加，因子组合的数目会急速增长。例如 2^3析因设计有 8 个组合，而 2^4析因设计则有 16 个组合。通常可以假定某些高阶相互作用的影响是可以忽略的，而主要影响和低阶交互影响可以通过运行部分或者完全析因实验获得，这种设计类型称为部分析因设计。

部分析因设计基于三个基本想法：

- 影响的稀疏性原理：很多情况下，系统是由某些主要和低阶交互影响驱动的。
- 投射属性：可以通过在部分析因设计中选择其重要因子的子集而获得一个较强的设计。
- 序贯实验：可以通过顺序运行两个或多个部分析因分析而获得一个较强的设计。

98
~
99

这些部分析因分析设计的主要用途是做筛查实验，实验的目的是识别对系统影响较大的因子，部分析因设计的例子有以下两种。

2^k 析因设计的 1/2 部分析因设计：选择 2^k 析因设计中 1/2 的组合，如果移除一个因子，则剩下的组合就是一个完整的 2^{k-1} 析因设计。见表 8-8。主体随机分配给选择的组合。在这个设计中有两个备选的切片，如果两个切片顺序运行，结果就是一个完整的 2^k 析因设计。

表 8-8　2^3 析因设计中一个 1/2 切片的例子

因子 A	因子 B	因子 C	主体
A1	B1	C2	2，3
A2	B1	C1	1，8
A1	B2	C1	5，6
A2	B2	C2	4，7

2^k 析因设计的 1/4 部分析因设计：选择 2^k 析因设计中 1/4 的组合，如果移除两个因子，则剩下的组合就是一个完整的 2^{k-2} 析因设计。见表 8-9。主体随机分配给选择的组合。在这个设计中有两个备选的切片，如果两个切片顺序运行，结果就是一个完整的 2^k 析因设计。不过，由于不是一个完整的析因设计，1/4 设计中的因子间可能存在依赖关系。

表 8-9　2^5 析因设计中一个 1/4 切片的例子

因子 A	因子 B	因子 C	因子 D	因子 E	主体
A1	B1	C1	D2	E2	3，16
A2	B1	C1	D1	E1	7，9
A1	B2	C1	D1	E2	1，4
A2	B2	C1	D2	E1	8，10
A1	B1	C2	D2	E1	5，12
A2	B1	C2	D1	E2	2，6
A1	B2	C2	D1	E1	11，15
A2	B2	C2	D2	E2	13，14

例如，在表 8-9，因子 D 依赖于因子 A 和 B 的组合，可以看出所有 A1 和 B1 的组合都有 D2，以此类推。同样，因子 E 依赖于因子 A 和 C 的组合。因此，如果移除 C 和 E（或者 B 和 D），则会变成两个重复的 2^{3-1} 析因设计，而不是一个 2^3 析因设计。
100
如果移除 D 和 E，则可获得后者。在表 8-9 中可以看到，如果移除 C 和 E，前四行的组合和后四行是等价的，由此可以识别出两个重复的 2^2 析因设计。

主体随机分配给选择的组合，这个设计中有四个备选切片，如果四个都顺序使用，其结果就是一个完整的 2^k 析因设计。如果两个切片顺序使用，就是一个 1/2 析因设计。

关于部分析因设计的更多详细内容见 Montgomery [125]。

总之，正确的实验设计选择是至关重要的，糟糕的设计无疑会影响得到正确的实验结论的可能性。而且，设计会设置一些采用统计方法的限制。最后要强调的是，要尽量使用简单的设计，并且尽最大可能地使用可用的主体，这一点非常重要。

8.6 实验工具

实验需要的工具一般有三种类型：对象、指南和度量工具。在计划阶段应该选择需要的辅助工具，并且特殊的实验工具应在实验前开发出来。

实验对象可以是规格说明书、代码文档等。做实验计划时，一个很重要的工作就是选择合适的实验对象，例如在一个关于审查的实验中，被检对象的缺陷数必须是已知的。可以植入缺陷或者用已知缺陷数目的文档。例如，可以用一个已知缺陷的、真实的早期版本的文档。

指南用于指导参加人员进行实验，一般包括过程描述、检查单等。如果实验中要比较不同方法，那么说明如何使用这些方法的指南也需要提前准备。此外，参与者还应该接受使用这些方法的培训。

度量是收集数据时进行的，在人力密集型实验中，数据通常通过人工或者访谈的方式获得。计划任务包括准备相关的表格和访谈提问单，并且要确认这些表格和提问单适用于实验参与者不同的背景和技能。练习中使用了一个收集主体经验信息的表格样例，见附录 A 中的表 A-1。

实验辅助工具的一般目标是为执行和监控实验提供手段和方法，缺乏合适工具的实验可能失控。无论这些工具怎样使用，实验结果都应该是独立且相同的。如果工具 101
影响到实验的效果，实验结果将是无效的。

实验验证将在 8.7 节详细介绍，工具准备的细节见 9.1.2 节和 9.2.2 节。

8.7 有效性评价

实验结果中备受关注的一个问题是实验的有效性。为了使实验结果充分有效，在计划阶段必须考虑与有效性相关的问题。充分有效指实验结果在利益群体的总体范围内有效。首先，结果应该在获得样本的群体内有效；其次，该结果有可能推广到更广泛的群体。当结果在我们期望的总体中有效时，该结果是充分有效的。

充分有效并不意味着最大范围有效。在某个组织内进行的实验可以设计为只针对该组织特有的问题，并且只要结果在特定的组织范围内有效就足够了。换句话说，如果要导出更加通用的结论，有效性也必须覆盖到更广的范畴。

对实验有效性的威胁有多种不同的分类方法。Campbell 和 Stanley 定义了两种类型：内部有效性和外部有效性威胁 [32]。Cook 和 Campbell 随后又扩展为四种威胁，分别是结论、内部、结构和外部有效性威胁 [37]。前一种分类在一些文献中有引用，但后一种更适用，因为它更容易映射到实验的各个步骤，见图 8-2。

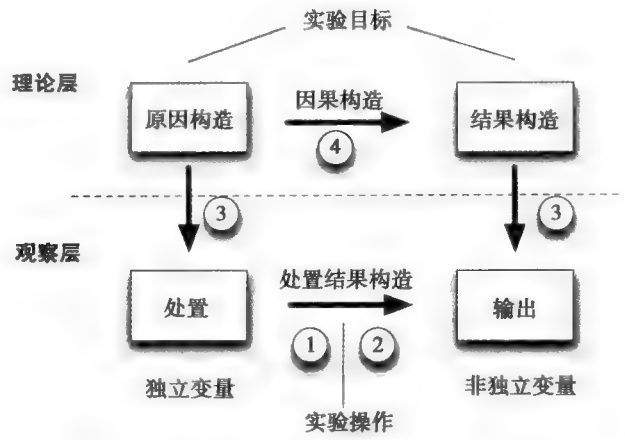


图 8-2　实验的原则（由 Trochim［171］改编）

Cook 和 Cambell 提出的四类威胁［37］都关联到实验中的方法问题。实验的基本原则见图 8-2。

顶层是理论领域，底层则是观察区域。实验的目的是希望基于观察到的现象导出在假设中定义的有关理论问题的结论。通过四步导出结论，而每一步都存在一类威胁，可能影响到结果的有效性。

（1）结论有效性。这里的有效性关心处置和处置结果之间的关系，期望在给定的显著性水平下，确保其统计关系存在。

（2）内部有效性。如果实验观察到处置和其结果之间的关系，我们必须确保这种关系确实是一种因果关系，而不是由于因子失控或者没有度量而造成的结果。换言之，确信该结果是该处置产生的。

（3）结构有效性。这里关心理论和观察之间的关系，如果是因果关系，则必须确保两件事：①处置良好地反映了原因的构造（见图 8-2 左部），并且②处置结果良好地反映了效果的构造（见图 8-2 右部）。

（4）外部有效性。外部有效性关心结论的通用性。如果在构造的原因和结果之间存在因果关系，那么研究结果是否在该研究范围之外也适用？是否在处置和结果之间存在关系？

结论有效性有时也称为统计结论有效性［37］，相对于定性分析的可靠性，见 5.4.3 节。对结论有效性的威胁，涉及实验得到的关于处置和其结果之间的关系是否正确的问题，如统计检验的选择、样本规模的选择、实现和度量时的小心谨慎等。

对内部有效性的威胁，主要关注是否认定了一些原本不存在的因果关系，影响内部有效性的因素包括：主体的选择和分类，实验中主体如何工作和获得酬金，实验中是否发生过一些特殊的事件等。这些因素都可能导致实验因受到干扰而表现出一些本不属于处置应该有的行为。

102
~
103

对结构有效性的威胁是指实验设置实际反映的所研究问题结构的程度。例如，如

果用大学里所修计算机科学课程的数目来度量主体在程序设计语言方面的经验，就可能是一个糟糕的度量，也就意味着结构有效性不好。如果用编程实践的年数来度量可能好一些，也就意味着其结构有效性好一些。

对外部有效性的威胁关心实验结果在实验设置以外的通用性。外部有效性受所选择的实验设计的影响，同时也受实验对象和主体选择的影响。一般有三种主要风险：选择错误的主体参加实验，在错误的环境开展实验，实验执行的时间可能对结果产生影响。

8.8 节给出了威胁有效性的详细列表。这个列表可以用作实验设计时的检查单。在有效性评价时，应该对照每一项检查是否存在威胁。对存在的威胁，我们可以解决也可以接受，事实上，有时我们不得不接受某些威胁。实现一个没有任何威胁的实验几乎是不可能的，因此，有些只能接受，但应在解释结果时说明。不同威胁类型之间的优先级将在 8.9 节进一步讨论。

8.8 有效性威胁的详细描述

下面，基于 Cook 和 Campbell［37］的研究，我们将列出对实验有效性的威胁并进行讨论。并非所有的威胁都适用于所有的实验，这个列表可以看成是一个检查单。表 8-10 总结了这些威胁，一种备选的不完全的分类方案见表 8-11。

表 8-10 Cook 和 Campbell［37］总结的有效性威胁

结论有效性	内部有效性
低统计效能 违反统计检验的假设条件 钓鱼与错误率 度量的可靠性 处置实现的可靠性 实验设置中的随机不相干性 主体的随机异构性	历史 成熟性 测试 工具 统计回归效应 甄选 死亡率 因果方向的歧义性 甄选互干扰 处置传播与模仿 处置补偿均等 补偿对抗 怨恨怠工
结构有效性	**外部有效性**
构造前解释不充分 单一操作偏倚 单一方法偏倚 结构和结构水平混淆 不同处置互扰 测试与处置互扰 结构间受限通用性 假设猜测 评价恐惧症 实验者预期	选择与处置互扰 设置与处置互扰 历史与处置互扰

表 8-11　Campbell 和 Stanley［32］总结的有效性威胁

内部有效性	外部有效性
历史	选择与处置互扰
成熟性	历史与处置互扰
测试	设置与处置互扰
工具	不同处置间互扰
统计回归效应	
甄选	

8.8.1　结论有效性

对结论有效性的威胁主要关注那些可能对导出正确结论造成影响的问题，这些结论是关于处置及其结果之间的关系的。

低统计效能。统计检验的效能表征一个检验可以从数据中揭示其真实模式的能力。如果效能低，则得出错误结论的风险就高，参见 8.2 节。更具体而言，也就不能拒绝一个错误的假设。

违反统计检验的假设条件。某些检验需要一些前提假设，譬如正态分布和独立样本。违反假定条件就可能导致错误的结论。有一些统计检验在前提假设方面具有相对较好的鲁棒性，具体见第 10 章。

钓鱼与错误率。这个威胁包括两个独立的部分。搜索或者钓鱼的办法会使得分析人员不再独立，而研究者会因为寻找指定的结果而影响实验结论。错误率关心实际的显著性水平。例如，在 0.05 的显著性水平上执行三个调查，整体的显著性水平就是 $1-(1-0.05)^3=0.14$。所以在执行多个分析时，应该调整错误率（显著性水平）。

104
~
105

度量的可靠性。实验的有效性高度依赖于度量的可靠性。传递下去，也就可能依赖于许多因素，比如糟糕的问题措辞，不好的实验工具或者工具布局。基本原则是，如果对某个现象做两次度量，两次的结果应该是一样的。举例而言，代码行比功能点度量可靠，因为它不需要人工判断。换言之，客观度量可以重复得到同样的结果，相比主观度量更可靠，参见第 3 章。

处置实现的可靠性。处置的实现意味着主体对处置的应用。如果不同的主体或者在不同场合应用处置的实现不是类似的，则会给实验带来风险。因此处置在不同场合、由不同主体的应用应该尽可能标准化。

实验设置中的随机不相干性。实验设置之外的一些因素可能会干扰结果，例如室外噪音、实验突然中断等。

主体的随机异构性。不同研究小组的异构总是存在的。如果异构严重，由于个体差异导致的变化就可能超过处置不同所产生的差异，这对实验结论也是个风险。另一方面，选择太同构的分组又会影响实验的外部有效性，见下面的讨论。例如，包含本科学生的实验会降低异构性，因为他们具有更加相似的知识和背景；但同时也降低了实验的外部有效性，因为这些主体不是从一个足够广泛的群体中选择的。

8.8.2 内部有效性

对内部有效性的威胁指可能在研究者不知情的情况下关乎因果关系中独立变量的风险，因此将威胁到处置及其结果之间因果关系的结论。有时，内部有效性威胁可以分为三类：单组威胁、多组威胁和社会威胁。

单组威胁（Single group thread）。这种威胁存在于只有一个组的实验中，未设计不采用该处置的对照组。那么，观察到的效果究竟是处置产生还是别的因素导致的？在决策时会有以下一些问题。

历史。实验中，可能会在不同的时间对同样的对象应用不同的处置，那么其风险是，时间历史会影响实验的结果，因为在两个时机的环境可能不一样。例如，第一个时机是节后第一天或者事情很少的一天，而另一个时机是正常的日子。

成熟性。这是主体随时间流逝具有不同反应的效果。例如主体会在实验中受到疲劳、枯燥等负面情绪的影响，或者在实验课程中受到学习等正面情绪的影响。 106

测试。如果重复测试，主体会在不同的时间给出不同的响应，因为他们已经知道测试是如何进行的。如果有必要熟悉测试，就不要将测试结果反馈给主体，这一点很重要，以免给主体不期望的学习。

工具。这是因实验中要用到的制品而引发的，如数据收集表、在审查类实验中的待检文档等。如果拙劣地设计这些工具，实验就会受到负面影响。

统计回归效应。当主体按照以前的实验或者案例研究进行分组，例如前十位、后十位，则可能产生这种威胁。这种情况下，即使根本不用处置，也应该进行一些增加或者改进。例如，由于纯粹随机变化的原因，以前实验的后十位在新的实验中未必全在后十位，这样选择的后十位可能不比剩下的后十位更糟糕。所以最好做一些改变，从相对更大的群体中选择。

甄选。这是由于人的行为会自然变化而产生的影响，从利益群体中选择主体的方法不同，选择的效果也是不同的。此外，让志愿者参与实验可能会影响实验结果。相对于整个群体，志愿者会更加主动并更适合新任务。因此，这样选择的主体不能代表期望的利益总体的情况。

死亡率。这种效应是由实验中各种人员的退出而引起的。刻画退出人员的特征，以检查是否依然可以代表总体样本。如果某个特殊类别的主体退出，例如审查相关实验中所有的高级评审者退出，实验的有效性就会受到严重影响。

因果方向的歧义性。这里的问题是，究竟是 A 导致 B，还是 B 导致 A，甚至 X 导致了 A 和 B？例如，如果程序复杂性和错误率之间有相关性，那么究竟是程序复杂性高引起了高的错误率，还是相反？或者是由于问题复杂性高导致了两者。

大多数对内部有效性的威胁都可以通过实验设计解决。例如，引入一个控制组可以解决许多内部威胁，但同时又引入了多组威胁。

多组威胁（Multiple group threat）。在一个多组实验中，要研究多个不同的组。对

这种研究的威胁是，控制组和选择的实验组可能会受到上述各种威胁的不同影响，出现甄选时的相互干扰。

107 *甄选互干扰*。甄选时的相互干扰源自在不同组中不同的行为表现。例如甄选成熟性干扰，意味着不同的组以不同的速度成熟，譬如两个组各自采用一种新方法，由于学习能力不同，一个组的学习速度比另外一个快，因此两个组的成熟性就不同。甄选历史表示不同组受历史的影响不同，以此类推。

社会性对内部有效性的威胁（Social threats to internal validity）。这些威胁在单组和多组实验都可能存在。下面给出的例子是一个审查实验，使用一个新的审查方法（基于视角的阅读，PBR）和老方法（基于检查单的阅读，CBR）进行比较。

处置传播与模仿。这种情况出现在控制组学习某组的处置，或者试图模仿这个组的行为。例如控制组使用 CBR 方法，实验组使用 PBR 方法，前者可能了解一些 PBR 方法，并且在进行审查时受到其视角的影响。对后者而言，如果评审人是某领域的专家，也会出现传播和模仿的威胁。

处置补偿均等。如果控制组没有处置，代之以获得某种补偿而作为控制组，这种情形会影响到实验的结果。例如教给控制组 BPR 之外的另外一种新方法作为补偿，他们的行为就会受到这种方法的影响。

补偿对抗。当主体可使用的处置少于期望时，这种天然的劣势可能激发主体的主观能动性，导致削弱甚至逆转期望的结果。例如使用传统方法的组可能尽最大努力表明老方法是有竞争力的。

怨恨怠工。这里和前一个威胁正好相反，当主体可使用的处置少于期望时，他们可能放弃而做不出正常应该有的效果。例如使用传统方法的组自暴自弃、消极怠工，而学习新东西的组会激励他们使用新方法。

8.8.3　结构有效性

结构有效性关注实验结果对于实验背后的概念或理论的普适性。对结构有效性的威胁一部分与实验的设计有关，其他是社会化的因素。

设计威胁（Design threat）。设计对结构有效性的威胁，涉及与设计有关的问题以及其反映研究结构的能力。

构造前解释不充分。尽管标题冗长，但这个威胁实际上非常简单，意指在将结构
108 转化成度量和处置前没有充分定义。理论不够清晰，于是实验也无法非常清楚。例如比较两个审查方法时，没有清楚地说明什么表示“更好”，是发现缺陷总数最多？每小时发现缺陷最多？还是发现最严重的缺陷？

单一操作偏倚。如果实验只有单一的独立变量、案例、主体或者处置，这个实验就可能只低度代表期望的结构而无法给出理论的全貌。例如一个审查实验只用单一文档作为实验对象，其原因结构的代表性就不够。

单一方法偏倚。使用单一的度量或观察类型带来的风险是，如果这个度量或观察

给出的结果有偏倚，就会误导实验结果。纳入多种不同的度量和观察类型，可以彼此进行交叉检验。例如在审查实验中，如果只度量发现的缺陷数目，而缺陷的分类却依赖于主观判断，那么发现的关系就不能充分地解释，这个实验可能就偏倚了该度量。

结构和结构水平混淆。如果一个关系中并不主要出现某个结构，但这个结构的水平却对该结果有重要影响，就意味着该结构出现的效果和结构水平的效果出现混淆。例如，以前是否有程序设计语言方面的知识不能解释为实验的原因，但主体具有1、3或者5年当前语言的经验却可能是实验结果出现差异的原因。

不同处置互扰。如果主体涉及多组研究，那么不同研究中的处置会相互干扰，进而无法推断一个效果来自某个处置，抑或一个处置的集合。

测试与处置互扰。测试本身也是处置的一次应用，可能会让主体对这个处置更加敏感或者更乐于接受。于是，测试就成为处置的一部分。例如测试中涉及度量编码中错误的数目，那么主体就会更容易意识到自己的错误，并尽量减少。

结构间受限通用性。一个处置可能正面地影响所研究的结构，但无意中，也可能负面地影响其他结构。这个威胁使结果难以推广到其他潜在的成果中。例如，一个比较研究推断采用新方法可以提高生产率，但另一方面可能观察到降低了可维护性，导致意料之外的副作用。如果没有度量或观察可维护性，就存在一个风险，即结论只基于生产率属性，而忽略了可维护性。

社会性对结构有效性的威胁。这些威胁关注与主体和实验员行为相关的问题。作为实验的一部分，他们可能发生与在其他地方不一样的行为，从而导致一个错误的结果。 109

假设猜测。人们参加实验时，通常会被告知实验的目标和预期的结果。这样他们可能会猜测假设，并基于对预期假设正面或负面的态度设立其行为的基调。

评价恐惧症。一些人害怕被评价。人类的一种倾向是，在被评价时总试图得到较好的结果，这会困扰对实验成果的评价。例如，比较不同估算模型时，人们可能不报告估算与结果的真实偏差，而是报告虚假但“好看”的值。

实验者预期。基于对结果的预期，实验会有意或无意地有所偏倚。这种威胁可以通过纳入不同期望的人员来缓解。例如，为了获得期望的答案，可以用不同的方式提问。

8.8.4 外部有效性

对外部有效性的威胁是一些影响实验结果推广到产业实践的限制条件。包括处置与人、地点、时间之间相互干扰的三种类型。

选择与处置互扰。这种情况指选择主体的群体不是我们期望要推广的群体，也就是说错误地选择了参与实验的人员。例如，对于程序员、测试人员以及系统工程师都会参与的审查，却只选择程序员参加实验。

设置与处置互扰。这种情况指实验设置和材料不具有代表性，如不代表工业实践。

例如，当现代工具已经在业界普遍使用时，实验中却使用一种老旧的工具。另外一个例子是在“过家家”的问题上做实验。这意味着错误的场地或者环境。

历史与处置互扰。这种情况指在某个特殊的、可能会影响结果的时间或日期进行实验。例如，如果在一个与软件相关的大崩盘后马上进行安全关键系统的调查问卷，较之于之前几天或者之后数周甚至数月，人们往往会给出不同的答案。

可以通过让实验环境尽可能接近现实情况来减少对外部有效性的威胁。但另一方面，现实也不是同质的。重要的是要刻画环境的特征，并报告所刻画的环境，譬如职
110 员的经验、工具、方法等，以评价其在特定情境中的适用性。

8.9　有效性威胁类型的优先级

前面讨论了内部有效性、外部有效性、结论有效性和结构有效性四种类型。一些对有效性的威胁彼此间会有冲突，某种类型的威胁上升了，另外一种就可能下降。因此，有效性类型的优先级排序问题实际上是在给定实验目标下的优化问题。

例如，在审查实验中用大学生，可以有机会获得较大的实验组，降低组之间的异构性，并实现可靠的处置。这样会有较高的结论有效性，但同时会降低外部有效性，因为如果我们希望结果可以推广到产业界，主体的选择就不具有代表性。

另外一个例子是让主体通过填写设计好的图表来度量某些因子，以便确保处置及其结果确实代表了所研究的结构。这样会提高结构的有效性，但却有可能降低结论的有效性。这是因为冗长的度量存在降低度量可靠性的倾向。

不同实验中，各类有效性的优先级也可能不一样，这取决于实验的目的。Cook 和 Campbell［37］针对理论检验和应用型研究给出如下建议。

理论检验（Theory testing）。在理论检验中，最重要的是检验是否存在某种因果关系（内部有效性），实验的变量代表理论的结构（结构有效性），增加实验规模通常可以解决统计显著性的问题（结论有效性）。理论很少与特定的设置、实验群体或者实验结果产生的时间有关联，所以几乎没有外部有效性的问题。因此，理论检验实验有效性的优先级降序排列为内部、结构、结论和外部。

应用研究（Applied research）。应用研究是大多数软件工程实验的研究目标，其考虑有效性的优先级也有所不同。同样，研究的关系（内部有效性）依然是第一位的，因为实验的主要目标是研究原因和结果之间的关系。在应用研究中，通用性——亦即从实验情境扩展到更加广泛的情境（外部有效性），也具有较高的优先级。对研究者而言，其兴趣不只在于看到实验结果在公司 X 有效，而是希望看到在特定规模或者领域的公司都有效。第三，相对而言，研究者对复杂处置中究竟哪个组件真正导致了结果（结构有效性）并不是很关心。例如在阅读实验中，不那么关心是由于评审人员提高了理解能力，还是具体的阅读程序帮助读者发现了更多的缺陷，其主要的兴趣在于效果本身。最后，在实际的实验设置中，很难得到充分的数据集，因此得出统计结论
111 的置信水平可能较低（结论有效性）。所以，应用研究实验有效性的优先级降序排列

为内部、外部、结构、结论。

总而言之，在实验的计划阶段，应该认真地评估和平衡对结果有效性的威胁。实验目的不同，评估有效性的优先级也不同。实验威胁对实验结果的实际意义影响重大，例如，统计上可能表明有意义，但实际上却是没有意义。这个问题将在 10.3.14 节仔细讨论。

8.10 实验举例

本节将继续 7.2 节介绍的例子，计划阶段的输入是目标定义。与计划相关的某些随目标定义已部分解决的问题在例子中已经阐述，学生将作为实验主体，且实验会包含多个需求文档。计划是实验的关键活动，计划时的失误会影响实验的整体效果。计划包含 7 个活动，见图 8-1。

情境选择。在许多案例中，情境类型至少部分取决于目标定义的方法。本例子隐含的实际情境应该是离线的，尽管有部分学生项目在线运行，但不能算作工业开发项目的一部分。实验由硕士和博士研究生混合进行。

一个由学生进行的离线实验，意味着可能很难有时间去审查一个正式的真实系统的需求文档。大多数情况下，这类实验不得不凭借特征有限的需求文档。在这个例子中，使用了两个来自实验包的需求文档（实验包包含一组可以重复使用的资料，并且可在线获得）。我们可以回忆一下，使用两个需求文档也影响设计类型的选择。由于需求文档在特征上有一些限制，所以它们可能在一定程度上被视为“玩具”需求文档。

如果实验的目标是单纯地（从研究者的视角）比较两种阅读技术，而不是比较公司内已经使用的阅读技术和新的备选技术，那么这个实验一般可以认为是有意义。对后者而言，应该让实验环境更接近公司的具体情况。但两者都必须确保公平比较。

在一般的研究案例中，公平比较才能支持被调查的两种技术是可比较的，这一点 112
非常重要。譬如很容易出现的情况是，在 CBR 方法中用了一个很糟糕的检查单，但另一方面给 PBR 方法提供很好的支持。这样会偏袒 PBR，导致实验结果受到质疑。这也是为什么“不支持”不是一个好的控制。一个实验的比较/评价必须在两个方法可比较且均获得相似支持的基础上进行。要避免对控制组不支持的情况发生。这种情况只有在获得支持的组比未获支持的组，如使用公司的老方法的效果还差时才有意义。然而，这种情况很少见，在这种环境下开展实验也几乎没有价值。

在这个具体的案例中，对双方支持的公平性是没有问题的，因为只要是将一个已经存在的技术和一个备选的技术对比，那么对两个技术的支持应该是公平的。这个案例的主要挑战在于，主体对已有技术是熟知的，而新技术是必须要教给他们的。所以对新技术的生疏是它的弱势。但另一方面，主体对新技术的兴趣又可能成为潜在的优势，因为他们可以学到新技术。因此在这种情况下，优劣并不是很明显，但是否存在对某个方法的偏爱却是研究人员必须要考虑的。

假设构建。目标定义时已经表达了实验希望用两种技术进行审查，并比较它们发

现故障的效率和效果。第一种方法是基于视角的阅读（PBR）审查，第二种方法是基于检查单的阅读（CBR）审查。PBR 基于评审人员在审查时不同的视角，CBR 基于一个检查单，其中列出了可能与需求文档缺陷有关的各种条目。

实验中使用的需求文档在之前的实验中已经用过，这意味着需求文档中的缺陷数目是已知的，尽管不能否认会发现新的缺陷。还应该注意到，效果指发现的缺陷在总缺陷中的占比，而效率还包括时间，亦即是否在单位时间内发现更多的缺陷。为了正式地构建假设，设 N 为缺陷数，Nt 为每个时间单元发现的缺陷数。

如下：

- $\mu_{N_{PBR}}$ 和 $\mu_{N_{CBR}}$ 分别表示 PBR 和 CBR 发现的缺陷数。
- $\mu_{Nt_{PBR}}$ 和 $\mu_{Nt_{CBR}}$ 分别表示 PBR 和 CBR 在每个单位时间发现的缺陷数。

假设可以描述为：

效果：

H_0：$\mu_{N_{PBR}} = \mu_{N_{CBR}}$

H_1：$\mu_{N_{PBR}} \langle\rangle \mu_{N_{CBR}}$

应该注意到，备择假设是两个阅读技术存在差异。换句话说，备择假设被描述为一个双面假设，而不是假定一个会好于另外一个。

效率：

H_0：$\mu_{Nt_{PBR}} = \mu_{Nt_{CBR}}$

H_1：$\mu_{Nt_{PBR}} \langle\rangle \mu_{Nt_{CBR}}$

这个假设意味着我们希望在统计意义上看到两个阅读技术发现的缺陷数不同，并且在单位时间内发现的缺陷数也不同，我们希望否定原假设。必须指出的是，不能否定原假设并不意味着接受原假设，它可能是由于主体太少而导致，而不是两个技术真的在发现缺陷的反应上一样。

变量选择。这个实验的独立变量是阅读技术，有两个水平，分别是 PBR 和 CBR。非独立变量是发现的缺陷数和单位时间发现的缺陷数。因此要保证主体可以清楚地标记发现的缺陷，研究者随后比较实验标记的缺陷和已知的缺陷集合。进一步还要确保主体记录时间，并填写发现缺陷的时间。必须注意的是，跟踪记录每个缺陷发现的时间非常重要，因为有的缺陷可能是误报的，所以这部分数据必须从相关的时间区间中去除。

主体甄选。最好的情况是有可能为实验随机地选择主体。然而，大多数实验的研究者往往被迫只能使用可用的主体，所以在大学进行的实验中，往往是选修某门课程的学生成为主体，本案例也是这样。在这种情况下，重要的是允许主体有拒绝参加的自由，而且不会受到任何惩罚。如果参与实验有学分，则应该提供其他可以不参加的备选方案。

如果实验的目的是比较两组学生使用不同技术的效果，那么应该通过主体的选择，亦即学生组的刻画来控制实验处置。事实上，这可以看作一个准实验。独立地刻画所

选择的主体非常重要，可以帮助我们评价研究的外部有效性。

设计类型选择。一旦知道了哪些人员将参与实验，下一步就是随机化主体选择并确定如何分组。好的方法通常先用一个预备测试来了解备选主体的经验并以此将人员分组，然后随机地从各组选择主体参加实验。这是为了确保每个组尽可能有同样的先 114
前经验，并维护主体的随机性。这也称为分块阻断，亦即阻断主体的先前经验以努力保证不影响实验的效果。最后，在大多数实验中，各组的大小规模一致，也就是说希望达到均衡设计。设计类型的选择会受到可用主体数目的影响。如果可用的主体数目较多，就可以考虑多种实验组合，或者每个/组主体只用在一个处置中。当可用主体相对较少时，如何巧妙地在不妥协实验目标的前提下地使用主体是很有挑战性的。

下一步是决定设计类型。这个实验包含一个主要影响因子（阅读技术）和两个处置（PBR 和 CBR）。第二个影响因子是需求文档，但不是实验真正关心的因素。基于前面的决策，自然地，设计类型是完全随机设计，每个组先使用 PBR 或者 CBR 阅读第一个需求文档，然后用另一种技术阅读另一个需求文档。决定实验顺序时有两个选择：①两个组先各用不同的阅读技术审查同一个需求文档，然后交换阅读技术审查另外一个需求文档；或者②两个组先用同样的阅读技术审查不同的需求文档，然后交换文档，并用另外一种技术。哪个方式都有顺序问题。第一种方式中一种需求文档会先于另一种被使用，而在第二种方式中一种阅读技术会先于另一种被使用。因此，必须考虑哪一种对实验的威胁最小。有效性威胁将在下面进一步讨论。

另外一种设计选择是允许一组用 PBR 审查需求文档，而另一组用 CBR 审查同样的文档。其优点是在同样的时间范围内可以使用较大的需求文档，缺点是只产生了一半的数据。一个实验常常只有有限确定的时间，因此，如何最有效地利用这个时间、最大可能地获得好的实验结果来处理假设变得非常重要。设计选择非常重要而且总是需要权衡取舍，不同的设计类型有不同的优缺点。此外，设计选择也是统计方法应用的基础，将在 10.4 节进一步讨论。

在这个具体的案例中选择了完全随机设计。一个组先使用 PBR 审查第一个需求文档，同时另一个组使用 CBR 审查同一个需求文档。选择这个方案的原因是，研究者相信阅读技术的顺序会比文档顺序的影响大，特别是这个实验的主要关注点是阅读技术之间的差异，而不是需求文档之间的差异。

实验工具。本实验基于一个实验包，需求文档是现成的，还有已知缺陷（到目前 115
为止）的清单。在确定实验将使用的需求文档时，最好已知其缺陷，以便于确定阅读技术的审查效果。

必须开发或者重用使用这两种技术的指南。特别重要的是要保证对比的公平性，如前面提到的，必须保证对两种技术的支持是可比的。

记录缺陷的表格也必须事先开发出来或者重用其他实验的，这里的关键是要保证需求文档和表格之间的可跟踪性，例如在需求文档标号发现的缺陷，在表格上要能获得该标号缺陷的相关信息。

有效性评价。最后，必须评价对有效性的威胁。提前考虑和排除风险非常重要，这样才能保证把威胁降低到最小程度。要避免所有的威胁几乎不可能，话虽如此，仍然希望在有可能的情况下识别所有威胁并尽可能缓解。

对这个例子中威胁的评价留在练习中，见 8.11 节中的练习 8.5。

实验过程中的下一步。基于上诉对本例子各步骤的介绍，希望我们已经准备好，可以进行这个实验了。然而，在执行之前，建议找一些同事评审一下实验设计。进一步地，如果有可能先做个简短的试验则更好，尽管试验要用到本来可以用作实验主体的一人或多人。也正因为如此，灵巧地使用潜在的主体非常重要。

8.11　练习

8.1　什么是原假设，什么是备择假设？

8.2　I 类错误和 II 类错误分别是什么？哪种更糟糕？为什么？

8.3　主体抽样有哪些不同的方式？

8.4　有哪些不同类型的实验设计？实验设计与分析时使用的统计方法有何关联？

8.5　在 8.10 节的例子中存在哪些威胁（考虑所有四种有效性威胁）？请解释它们为什么是威胁？如何在这些不同的有效性类型中平衡取舍？

116

第9章
Experimentation in Software Engineering

操　作

设计和计划好之后，就要实施实验以获取数据进行分析，也就是这里要讨论的实验操作。在实验的操作阶段，主体应用处置，这也意味着这个阶段是主体真正使用处置的地方。在大多数软件工程实验中，其他阶段仅会在很短的时间里真正涉及主体，涉及主体的事件通常包括主体承诺参加实验前的说明会、实验完成时的报告会等。尽管有时会进行在2.4节讨论的面向技术的实验，但大多数情况下，软件工程的实验都是和人打交道的。本章在一定程度上涉及一些如何激励人们参与实验的讨论。

即使实验的设计很精心，收集的数据也以合适的方法进行了分析，但如果主体没有认真地对待实验，结果也将是无效的。实验心理学领域对涉及人的实验进行了一些研究［4，29］，来自于这个领域的实验实施指南在一定程度上也适用于软件工程领域。

实验操作阶段包括三个步骤：准备——选择主体并准备需要的表格；执行——主体以不同的处置方式执行任务并收集数据；数据确认——确认收集到的数据。这三个步骤如图9-1所示，并将在本章后续部分进一步介绍。

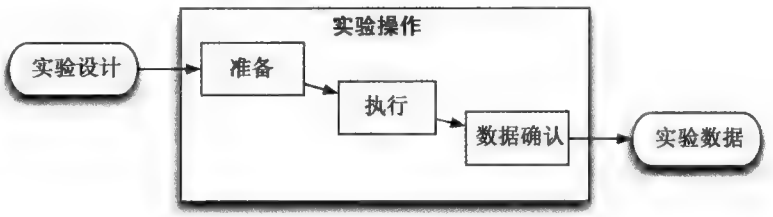


图9-1　实验操作的三个步骤

9.1　准备

实验真正开始前还有一些准备工作，准备越充分，执行越容易。准备工作有两个重要的方面。首先是选择和通知主体；其次是准备实验需要的材料，如表格、工具等。

9.1.1　参与者承诺

实验开始前，要找到作为主体参与实验的人员，让这些人员有参与到整个实验的激情和愿望是至关重要的。

在许多情况下，应该寻找那些其日常工作和实验要开展的任务类似的人员。例如，如果某个实验涉及用不同的工具编写C代码，选择平常写C代码而不是Java代码的人

员就更合理。如果选择的人员不能代表我们希望能够展示结果的群体，就会威胁到实验的外部有效性。见第 8 章，8.4 节讨论主体选择部分的抽样技术。

找到正确的人员后，要说服他们参加实验，同时要考虑这些人作为实验主体是否存在职业伦理方面的隐患。

获得同意（obtain consent）。参与者必须同意研究目标。如果参与者不了解工作的意图，或者与他们认为其所同意的工作相去甚远，他们就有可能不按照目标和应有的能力执行实验，导致数据无效，从而给实验带来风险。清楚地描述如何使用和公开实验结果非常重要，还要让参与者清楚地知道他们可以自由地退出实验。有的时候，必须在关乎有效性的设计和这里讨论的问题之间进行折衷。如果参与者会受到实验的影响，就会影响到实验的有效性。

敏感结果（Sensitive result）。如果实验结果会影响参与者，也就是说对参与者敏感，则必须保证有关他们个人表现的实验结果一定要保密，这一点很重要。有的时候，很难判断结果是否敏感，但通常只要结果在实验之外可能对参与者有指征，亦即可能对应到某个参与者，则该结果就可以认为有一定的敏感性。例如，如果实验度量程序员的生产率，其结果就可能指征该程序员的能力，那么这个结果就应该是敏感的。反之，如果要求参与者在验收测试中使用某种方法，而他平时从不会涉及这类测试，那么这个实验结果就可能不是敏感的。

诱惑（Inducement）。吸引人们参与实验时通常会提供某些奖励以诱导人们积极参加，但这种诱惑不能过大，那样可能导致人们仅仅是因为受到诱惑而参与实验，而不会认真地对待实验。

披露（Disclosure）。披露指尽可能开放地对实验主体展现实验的细节。反之，蒙蔽或出卖参与者通常是不可接受的。这种情况下，如果有备选方案，应该使用备选的方法；如果不能披露是唯一的方案，则只能用于所隐蔽的方面对参与者无关紧要的情况下，并且不会影响参与者参加实验的意愿。对于部分披露的情况，要尽可能早地向参与者解释清楚。

更多关于实验伦理方面的讨论见 2.11 节。

9.1.2　准备实验工具

执行实验前，所有需要的实验工具都要就绪，见 8.6 节。包括实验对象、实验指南、度量工具和表格等。需要的工具是由实验设计以及数据收集方法决定的。

如果主体自己要采集数据，在大多数情况下意味着必须给他们提供一些表格。在构造表格时应确定参与者具名还是匿名填写表格。如果没有另外的研究需求，并且没有区分参与者的实际意义，使用匿名表格可能更加合适。不过这也意味着，如果某些内容填写得不清楚，也无法联系到参与者。

在许多情况下，设计都会考虑随机和重复检验，不同的参与者会作为不同处置的主体，所以应该为每个参与者准备一套个性化的工具集。当参与者匿名时也可以这

样做。

如果用访谈方式采集数据，在实验前要准备提问单。同样，也应该为不同的参与者准备不同的提问单。 [119]

9.2 执行

实验可以以许多不同的方式执行。某些实验，诸如简单审查实验可以将所有参与者集中在一个场合进行，如会议。其优点是数据采集的结果可以直接在会议上获得，而不再需要随后联系参与者分别征询结果。另一个优点是实验人员也在会场，如果有问题可以直接解决。

然而，有些实验是在一个相当长的时间跨度内执行的，实验人员不可能参与到实验和数据采集的每一个细节，例如实验关联到一个或者多个大项目，希望评价这些项目使用的不同的开发方法。Ohlsson 和 Wohlin[128] 给出过这样的一个案例。在一个历时两年有关大规模软件开发的课程中，每一年大约 120 名学生并行运行 7 个项目，Ohlsson 和 Wohlin 实验 [128] 的目标是评价收集工作量数据时，各种技术所采用的形式化方法的程度。

9.2.1 数据收集

数据可以通过多种方式收集，譬如参与者手工填写表格、工具支持下的手工收集、访谈或者工具自动收集。

使用表格的优点是无需实验人员太多的工作量，因为实验人员不必完全参与收集活动。其缺点是实验人员不能直接发现表格中可能存在的不一致、不确定、疏漏等缺陷。这类缺陷在参与者发现或提出疑问甚至在完成数据收集之前都不易发现。访谈的优点是实验人员有可能和参与者进行良好的沟通，缺点当然是需要实验人员更多的工作量。

9.2.2 实验环境

如果实验是在一个常规开发项目中进行，则需要避免对项目不必要的影响。这是因为在项目中进行实验的目的是观察在项目环境下不同处置的效果，如果因为实验而导致项目环境有过大变化，则会影响真实的效果。 [120]

不过在某些情况下，实验和项目有一定的交互也是有益的。例如实验发现项目的某些部分可以执行得更好或者估算不正确，把实验发现告诉项目领导应该是合适的。这些来自于实验的直接反馈有助于激励项目人员参与实验。

9.3 数据确认

数据采集后，实验人员必须检查以确定数据是否合理以及是否是正确采集的，譬如参与者是否理解表格并正确填写了数据等。另外一种错误是参与者可能没有认真地

进行实验，这部分数据也应该在分析前从数据集中移除。数据异常分析将在 10.2 节进一步讨论。

评审以确保实验以预期的方式真实地执行非常重要，譬如主体是否按正确的顺序应用正确的处置。如果出现差错，则无疑数据是无效的。

召开研讨会或者提供其他某种展示数据收集结果的方法，是检查参与者是否理解实验意图的有效途径。这提供了一种机会，让参与者可能反思那些他们不同意的结果，也有助于建立长期的信任。具体见 2.11 节。

9.4　操作举例

以 8.10 节介绍的实验设计作为操作的输入，包含三个必需的步骤。

准备。首先要确定主体。在这个例子中，邀请博士和硕士学生作为主体。一旦有了潜在的参与者集合，说服他们参加实验并获得其承诺是非常重要的。初步承诺后，必须确保参与者同意。建议使用知情通知书，即便不一定需要正式的条款。其他要考虑的就是与实验伦理有关的问题，见 9.1.1 节。必须随机地为各个处置分配主体。如
121 果设计包括分块因子（如学生类型），主体应该先按因子分块，然后再在每个块组中随机指派到各个处置。如果是平衡设计，则应该为每个组选择同样数目的主体。

下一步要确保需要的基础设施就位。包括合适的房间，譬如房间应该给主体之间留出足够的距离；供所有主体使用的实验文档和表格副本；房间中的钟表，供主体记录收集数据花费的时间，不能假定每个人都用自己的时钟；等等。

执行。实验中很重要的是要确保人们合适地分散在房间中。做审查实验时，有可能在一次实验中让所有主体在同一时间进行审查。这也意味着方便为实验中的任何问题提供支持。根据实验中采用手工方式还是用计算机填表，也应做好相应的工具准备。

数据确认。最后，收集的数据必须得到确认。有的时候，一些主体很早结束并离开，他们完成的表格必须仔细检查以确保他们是以合理的方式填写了表格。此外，必须检查每个人都理解了正确填写数据的方式，否则必须移除一些未正确收集的数据。

9.5　练习

9.1　在选择主体时，应该考虑哪些因素？

9.2　为什么在实验中伦理问题是重要的？

9.3　为什么必须在实验前精心准备必需的实验工具？

9.4　数据确认是什么？为什么需要在统计分析之前进行数据确认？

122 9.5　在实验结果中涉及个人利益时，如何与主体沟通？或者应该如何处理？

第10章
Experimentation in Software Engineering

分析与解释

从操作阶段获得的实验数据将作为分析与解释阶段的输入。在操作阶段收集了实验数据之后，我们希望通过这些数据能够推导出一些结论。而为了能够推导出有效的结论，则必须解释实验数据。定量解释可以通过如图 10-1 中所示的三个阶段来实施。

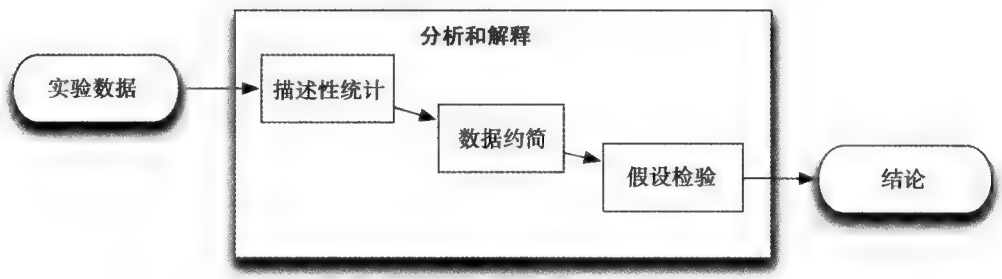


图 10-1　定量解释的三个步骤

第一阶段，采用描述性统计分析数据特征，包括可视化居中趋势、离散程度等；第二阶段，移除异常数据或错误数据，即将数据集约简成有效数据点集；第三阶段，使用假设检验分析数据，并在给定的显著性水平下统计评估实验假设。后续章节将对这几个阶段进行更详细的描述。

10.1　描述性统计

描述性统计常用于数据集的展示和数值处理。收集了实验数据之后，可利用描述性统计将数据集的某些方面进行描述和图形化展示，如在某种尺度下的数据展示、数据集的集中程度或分散程度如何等。描述性统计的目标是描述数据集是如何分布的。为了更好地理解数据的性质、识别出异常数据或错误数据（也称作离群点），描述性统计可能会在执行假设检验之前进行。

本节将介绍一些描述性统计和绘图技术，以帮助我们了解一个数据集的概貌。度量的尺度（见第 3 章）限制了统计的类型，而统计类型又会影响计算结果。表 10-1 总结了一些统计学方法及这些方法可以使用的尺度。值得注意的是，如表 10-1 所示，某种尺度类型的度量手段可以被多种尺度使用，如众数（mode）在四种尺度中均可使用。

表 10-1　每种尺度所对应的一些相关统计

尺度类型	居中趋势度量	离散性	依赖关系
定类尺度	众数	频率	
定序尺度	中位数、百分位数	变化间隔	Spearman 相关系数 Kendall 相关系数
定距尺度	均值、方差和值域	标准差	Pearson 相关系数
定比尺度	几何平均数	变异系数	

10.1.1　居中趋势的度量

居中趋势的度量指标（如均值、中位数、众数）表征了一个数据集的“中间”。如果通过对随机变量采样获得了数据集中的数据点，那么统计获得的这个中间点（通常称作平均数）就可以被解释为对这个随机变量的期望的估计。

在描述居中趋势度量中，假设对某个随机变量进行采样，获得 n 个数据点 x_1，…，x_n。其（算术）均值 $\bar{x}$ 的计算方式如下：

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

均值适用于定距尺度和定比尺度的居中趋势度量。例如，数据集（1，1，2，4）的均值根据上式计算可得：$\bar{x} = 2.0$。

中位数 $\tilde{x}$ 表示一个数据集中处于中间位置的数据值，也就是说样本中大于中位数的样本数目与小于中位数的样本数目相同。中位数可以通过对样本进行升序或者降序排列，然后挑选出中间位置的样本来计算。如果 n 是奇数，很明显中位数就是中间位置的数；如果 n 为偶数，中位数取中间两个数值的算术平均数。计算算数平均数的操作要求尺度至少是等距的。如果尺度是顺序类型的，则可以通过随机选择其中一个或者用一个中间值对来表示中位数。

124

中位数适用于类型为定序尺度、定距尺度和定比尺度的数据。例如，我们可以计算出数据集（1，1，2，4）的中位数为 $\tilde{x} = 1.5$。

中位数是百分位数的一个特殊情况，也就是位置在 50% 的数，定义为 $x_{50\%}$，表示有 50% 的样本位于 $x_{50\%}$ 之下。一般来说，百分位数 $x_{p\%}$ 表示样本中有 $p\%$ 的样本数据位于这个值之下。百分位数对于类型为定序尺度、定距尺度和定比尺度的数据是有意义的。

众数代表最常出现的样本值。通过计算样本中每一个数值出现的次数并选择其中出现次数最多的值，就可以计算出众数。如果出现次数最高的值只有一个，则众数是显而易见的。如果一个样本中出现次数最高的值有奇数个，则可以用这奇数个值的中间值作为众数。而这第二个操作要求尺度至少是定序的；如果是定类尺度，则可以在出现次数最高的样本值中随机选择一个作为众数或者用这些最常见样本值构造值对来表示众数。

众数适用于描述定类尺度、定序尺度、定距尺度和定比尺度的数据。例如，数据集（1，1，2，4）的众数为 1。

几何平均数是一种不太常见的度量居中趋势的方法，其值为所有样本值乘积的 n 次方根，计算公式如下所示。

$$\sqrt[n]{\prod_{i=1}^{n} x_i}$$

如果所有样本都是非负的并且对定比尺度有意义，则几何平均数是良定义的。如图 10-2 所示，如果样本的分布是对称的，则其（算术）平均数和中位数相等。如果分布既是对称的又具有唯一的最大值，则度量居中趋势的这三个值都相等；如果样本的分布是偏态分布，则其均值、中位数和众数可能不同。

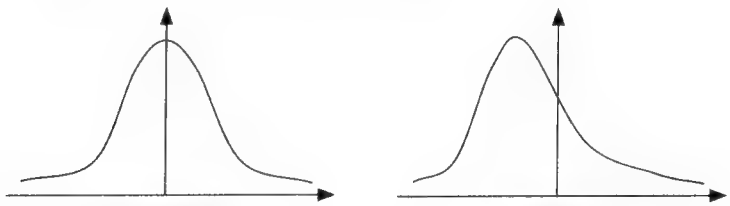


图 10-2　对称分布的均值、中位数和众数相同，非对称分布的均值、中位数和众数可能不同

例如，如果分布中的右尾部分长，则均值会增大，而中位数和众数并不受影响。这表明均值是一个更敏感的度量指标。然而，它要求数据至少是定序尺度的，因此，当尺度类型不满足条件时，就无法使用均值来度量居中趋势。 125

10.1.2　离散性的度量

居中趋势的度量并不能展示数据集的离散性。因此，我们还需要测量数据集偏离居中趋势的水平，即数据有多么集中或者有多么发散。（样本）*方差*是一种常用的离散性度量方法，记为 s^2，其计算方式如下：

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

因此，方差表示的是数据与样本均值的平方距离。公式中的分母是 $n-1$ 而不是 n，这看起来可能很奇怪，但是这样做可以使得方差得到一些期望的属性。具体而言，样本方差是随机变量方差的一致无偏估计量。方差对类型为定距尺度和定比尺度的数据有意义。

*标准差*记为 s，定义为方差的平方根：

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

标准差比方差更常用，因为它与数据值本身维度（度量单位）相同。标准差对类型为定距尺度和定比尺度的数据有意义。

数据集的值域（range）指数据中最大值和最小值之间的距离：

$$\text{range} = x_{\max} - x_{\min}$$

值域的值对类型为定距尺度和定比尺度的数据有意义。

变化区间（variation interval）用数值对（$x_{\min}$，$x_{\max}$）表示，其中包括数据集的最小值和最大值。变化区间对类型为定序尺度、定距尺度和定比尺度的数据有意义。

有时离散性可以用均值的百分比表示，即变异系数（coefficient of variation），公式如下：

$$\frac{s}{\bar{x}} \times 100\%$$

变异系数没有维度，它对于类型为定比尺度的数据有意义。

离散性一般而言可以通过每个数值出现的频率来表示。频率表就是通过列出每个不同值和其出现的次数的方法构造的。通过将每个频率与样本总数相除可以得到相对频率（relative frequency）。例如，对于一个有 13 个样本的数据集（1，1，1，2，2，3，4，4，4，5，6，6，7），可以创建如表 10-2 所示的频率表。频率对所有尺度类型
126 的数据来说都是有意义的。

表 10-2　频率表实例

数　值	频　数	相对频率
1	3	23%
2	2	15%
3	1	8%
4	3	23%
5	1	8%
6	2	15%
7	1	8%

10.1.3　依赖关系的度量

当数据集是由随机变量 X 和 Y 产生的关联样本（x_i，y_i）组成时，度量变量间的依赖关系通常是有意义的。

如果 X 和 Y 可以通过某函数 $y=f(x)$ 关联起来，就可以利用样本估计这个函数。如果假设函数 $y=f(x)$ 是线性的并且可以写成 $y=\alpha+\beta x$ 的形式，则可以使用线性回归法（linear regression）来估计这个函数。回归意指利用数据点来拟合某个曲线，在本案例中，我们将展示如何通过直线拟合来做线性回归，使得各数据点到该直线的平方距离之和最小。在给出公式之前，需要定义下列经常出现的“和”的简单表示：

$$S_{xx} = \sum_{i=1}^{n} (x_i - \bar{x})^2$$

$$S_{yy} = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

$$S_{xy} = \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) = (\sum_{i=1}^{n} x_i y_i) - \frac{1}{n}(\sum_{i=1}^{n} x_i)(\sum_{i=1}^{n} y_i)$$

这些“和”可用于计算回归线 $y = \bar{y} + \beta(x - \bar{x})$，其斜率为：

$$\beta = \frac{S_{xy}}{S_{xx}}$$

并且该直线与 y 轴交于点 $\alpha = \bar{y} - \beta\bar{x}$。

如果依赖关系不是线性的，则可能可以通过寻找一种数据转换将该关系转换为线性关系，再使用线性回归法。例如，如果关系是指数型的，$y = \alpha x^{\beta}$，这就意味着对数据进行对数转换可以得到线性关系 $\log(y) = \log(\alpha) + \beta\log(x)$。因此，在对数转换之后，我们就可以用线性回归计算该直线的参数。

127

度量两个数据集 x_i 和 y_i 间的差别有多大时，可以使用*协方差*（covariance）。协方差也是度量依赖关系的一种方法，记为 c_{xy}。其定义如下：

$$c_{xy} = \frac{S_{xy}}{n - 1}$$

协方差对类型为定距尺度和定比尺度的数据有意义。协方差取决于每个变量的方差，并且为了能比较不同相关变量间的依赖关系，协方差可被规范化为 x_i 和 y_i 的标准差。这样就可以得到*相关系数* r（correlation coefficient），也称为 Pearson *相关系数*。其计算公式如下：

$$r = \frac{c_{xy}}{s_x s_y} = \frac{S_{xy}}{\sqrt{S_{xx} S_{yy}}} = \frac{(n\sum_{i=1}^{n} x_i y_i) - (\sum_{i=1}^{n} x_i)(\sum_{i=1}^{n} y_i)}{\sqrt{(n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2)(n\sum_{i=1}^{n} y_i^2 - (\sum_{i=1}^{n} y_i)^2)}}$$

r 取值 -1 到 +1 之间。当数据间没有相关性时，r 值为 0；但是反过来是不成立的。因为即使当 $r = 0$ 时，x_i 和 y_i 也可能是非线性的强相关。（Pearson）相关系数只度量线性依赖，如果 x_i 和 y_i 的数据尺度类型是定距尺度或定比尺度时，这时计算其相关系数是有意义的。相关系数适合衡量呈正态分布的数据。

如果数据尺度类型是定序尺度或者数据不符合正态分布时，可以使用 Spearman *等级相关系数*（Spearman rank-order correlation coefficient），记为 r_s。Spearman 等级相关系数的计算方法与 Pearson 相关系数相似，使用秩（即样本排序后的序号）来代替样本值，例子可参见 Siegel 和 Castellan［157］。

另一种对依赖关系进行度量的方法是 Kendall *等级相关系数法*（Kendall rank-order correlation coefficient），记为 T。Kendall 等级相关系数和 Spearman 等级相关系数一样，适用于排序数据，即数据至少是成对有序的样本。然而 Kendall 等级相关系数的基础理论不同，它关注在样本排序中计数排序意见相同和不同的意见数量，例子可参见 Siegel 和 Castellan［157］。

如果变量超过两个，则可以使用多变量分析（multivariate analysis），包含多元回归（multiple regression）、主成分分析（principal component analysis，PCA）、聚类分析（cluster analysis）和判别式分析（discriminant analysis）。这些技术的介绍可参见各种统计学文献，如 Manly[118] 和 Kachigan[90，91]。

10.1.4　图形可视化

在描述一个数据集时，定量度量其居中趋势、离散性和依赖关系时，应该结合图形可视化技术。因为图形非常直观，能很好地呈现数据集的概貌。

128

散点图（scatter plot）是一种简单但有效的图示，如图 10-3所示，将成对的样本点根据其坐标值（x_i，y_i）绘制到二维坐标系内。

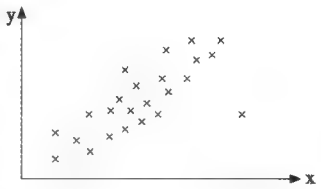


图 10-3　散点图

散点图可以用于评估变量之间的依赖关系。通过检查散点图，很容易发现数据是如何分布和聚集的，并由此判断数据之间是否有存在线性依赖关系的倾向。我们可以识别出不合规则的点（离群点）并观察到点与点之间的相关关系。图 10-3 中，数据点之间有一种正相关的线性趋势，我们也可从中发现潜在的离群点。在这个特例中，存在一个候选的离群点。

箱形图有助于我们观察样本的离散性和偏度（skewedness）。如图 10-4 所示，箱形图是通过对不同的百分位数进行图形化展示来构造的。箱形图有不同的制作方式，这里我们选用 Fenton 、Pfleeger 和 Frigge 等人［56，60］所倡议的方式来绘制。不同绘制方式的主要区别在于如何处理箱尾（whiskers）。有的文献（如 Montgomery[125]）认为，箱尾（上尾和下尾）应该使用数据集的最大值和最小值来表示，而 Fenton 和 Pfleeger[56] 则提议使用箱体长度的 1.5 倍分别加减数据集的上下四分位数来表示箱尾。

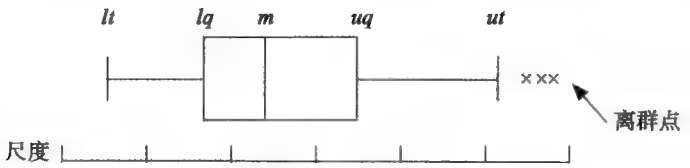


图 10-4　箱型图

箱体中间标为 m 的线代表数据集的中位数，下四分位数（lq）是指 25% 分位数（小于 m 的数据的中位数），上四分位数（uq）是指 75% 分位数（大于 m 的数据的中位数）。箱体的长度 $d = uq - lq$。

箱体的边缘（lt 和 ut）表示数据的理论边界。如果数据集是正态分布的，则所有数据都应该能在上边缘和下边缘之间找到。上边缘 ut 由 $uq + 1.5d$ 计算得到，下边缘 lt 则相应地由 $lq - 1.5d$ 计算得到［60］。为了避免无意义的值（例如负的代码行数），边

缘值应该截断到其最接近的实际值。 129

位于上边缘和下边缘之外的值被称为离群点，应该明确地在箱形图中展示出来。如图 10-4 所示，其中有三个离群点。

直方图（histogram）可以用来展示来自同一变量的样本的分布密度情况。如图 10-5所示，一个直方图由一组高低不同的矩形构成，矩形的高度代表一个值或者一个区间的出现频率（或者是相对频率），因此，直方图也是频度表的图形化表示。正态分布是我们应该特别关注的一个分布，因为我们进行数据分析时，必须考虑数据是否符合正态分布。直方图可以用来初步判断数据集是否近似于正态分布。当然也可以检验数据是否符合正态分布，这一点我们将在 10.3 节介绍卡方检验时进行深入讨论。

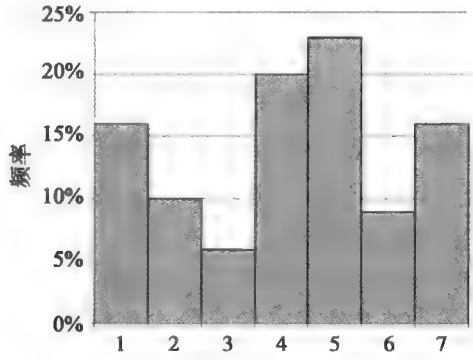


图 10-5　直方图

累积直方图（cumulative histogram）如图 10-6 所示，可以用于描述一个变量的样本的概率分布函数。图中每一个矩形表示到目前类别为止所有出现的频率值的累积和。

如图 10-7 所示，饼图（pie chart）用来描述将数据值分为若干特定类别后，各类别之间的相对频率。饼图将各个类别的片段组合成一个圆，其中每个扇形的角度与本类的相对频率成正比。

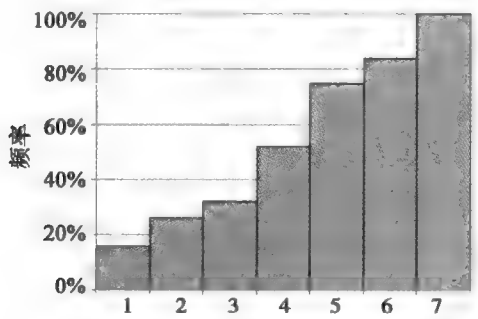


图 10-6　累积分布直方图

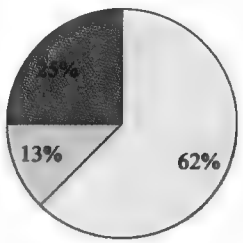


图 10-7　饼图

10.2　数据约简

10.3 节将介绍一系列的统计方法。这些方法都有一个共同点——它们的统计结

果高度依赖于输入数据的质量。如果统计方法使用的数据并不具备应有的性质，那么，利用这些方法的输出结果推导出的结论当然就不会正确。

数据集之中的错误可能是系统误差，也可能是离群点。离群点意味着这些数据点与其他数据点相比可能偏离期望值很多，如图 10-8 所示。

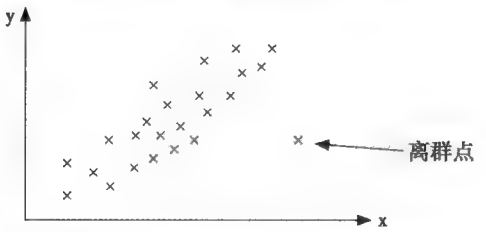


图 10-8　一个在散点图中被检测出的离群点

如图 10-8 所示，绘制散点图是一种识别离群点的有效途径。绘制如图 10-4 所示的箱形图也是一种途径。有一些统计学方法可以用来识别离群点。例如，假设数据集符合正态分布，则可以计算出某个值（例如最大值和最小值）属于该正态分布的概率。这可以通过计算可能的离群点同均值之间的差异，或计算离群点与其最近的相邻点之间的差异来进行。当发现差异较大时，测定出这些差异出现的概率。这种方法旨在评估那些发现的、看起来像是极值点的离群点是否确有可能来自于正态分布。

130
~
131

应该注意的是，此处谈论的数据约简同第 9 章讨论的数据确认有关。数据确认处理的是由实验执行不当而产生的虚假数据点，比如，判定人们是否认真参与实验。本节讨论的数据约简并不仅仅关注由于实验执行不当而产生的虚假数据点，还关注收集的数据分析得到的结果，例如采用描述性统计方法得到的结果。

确定了离群点以后，决定如何处理它们也很重要。处理并不仅仅意味着在图中标识这些数据，分析离群点的产生原因也很重要。如果离群点是由异常或偶然事件引起的，且此类事件不再发生，则可以将此类离群点移除。例如，如果这个点是完全错误的或者被曲解的，则可以移除它。

如果离群点的出现是由于偶然事件引发的，但该事件有可能再次发生，例如，如果一个模块是由一个没有经验的员工实现的，那么我们不建议将此离群点移除，因为此类离群点包含了太多的相关信息。如果离群点的出现归因于尚未考虑的变量，例如工作人员的经验，就应该将此变量纳入模型和计算中考虑。这也可能会派生出两种模型。就以这个员工经验的案例为例，这意味着可以将模型分为为正常员工建立的模型（其中离群点被移除）和为没有经验的员工建立的模型。对待离群点时应该一事一议。

除了需要将无效的数据从数据集中移除外，冗余数据有时也需要进行处理。如果冗余数据太多，有时会使得分析无效。因子分析和主成分分析（PCA）可以用于识别冗余数据。这些技术能够识别正交因子，从而替换原始因子。由于这方面的技术与本书的相关性不大，在此不再赘述，需要者可参见 Kachigan[90, 91] 和 Manly[118]。

10.3　假设检验

10.3.1　基本概念

假设检验的目的是确定是否能够基于某统计分布的一个样本拒绝某个原假设 H_0。也就是说，原假设描述了分布（用于抽取样本的分布）所具有的一些性质，实验者希望拒绝该假设，即在给定的显著性水平下这些性质为真是不成立的。原假设在第 8 章中已经进行过讨论。通常情况下，分布依赖于单个参数。设定了 H_0 就意味着明确描述了分布，并给将要被检验的参数赋了值。

例如，假如一个实验者观察到了一辆车，并且希望说明这辆车不是一辆小汽车。实验者知道所有的小汽车都有 4 个车轮，但他也知道除了小汽车之外也存在着具有 4 个车轮的车辆，此时，可建立一个非常简单的原假设："H_0：观察到的车辆是小汽车"。

为了检验 H_0，我们定义一个测试单元 t，并且给定一个临界区域 C。这个临界区域 C 是测试单元 t 值域中的一部分。这就意味着显著性检验可以表示为： 132

- 如果 $t \in C$，则拒绝 H_0。
- 如果 $t \notin C$，则不拒绝 H_0。

在本例中，测试单元 t 是车轮的个数，临界区域是 $C = 1, 2, 3, 5, 6, \cdots$，检验是假如 $t \leqslant 3$ 或者 $t \geqslant 5$ 则拒绝 H_0，否则不拒绝 H_0。

如果观察到 $t = 4$，即表示不能拒绝原假设，但也不能得出结论。这是因为除了小汽车之外还有其他车辆有 4 个车轮。

因此，原假设应该选择负面的陈述，也就是说，检验的目的是拒绝原假设。如果无法拒绝原假设，就无法从实验结果中推导出任何结论；假如能够拒绝原假设，则意味着在给定的显著性水平（α）下假设是假的，详见下文。进行检验后，往往能够计算出最低显著性水平（通常用 p-值来表示），表示该显著性水平下能够拒绝原假设。统计分析软件包通常会报告这个值。

临界区域 C 可能具有不同的形状，但它通常表现为区间，例如，$t \leqslant a$ 或 $t \geqslant b$。如果 C 包含一个这样的区间，则 C 是单边的；如果 C 包含两个区间（$t \leqslant a$，$t \geqslant b$，且 $a < b$），则 C 是双边的。

假设检验有如下三个重要概率：

$$\alpha = P(\text{Ⅰ 类错误}) = P(\text{拒绝 } H_0 \mid H_0 \text{ 为真})$$

$$\beta = P(\text{Ⅱ 类错误}) = P(\text{不拒绝 } H_0 \mid H_0 \text{ 为假})$$

$$\text{Power} = 1 - \beta = P(\text{拒绝 } H_0 \mid H_0 \text{ 为假})$$

这些概率已经在第 8 章中讨论过了。

这里，我们尝试以一个简单但很有说服力的检验（二项式检验）为例来说明上述概念。一个实验人员在测试一个产品时发现了若干错误，并将其分为两类："破坏型错

误”（会破坏程序数据的错误）和“非破坏型错误”（不会破坏程序数据的错误）。实验者的理论是：“非破坏型错误”比“破坏型错误”出现得更普遍。因此，实验者想要执行一个检验来看不同类型错误数目的差异是偶然造成的还是一种系统性的差异。

原假设即为测试发现一个“破坏型错误”和一个“非破坏型错误”的概率没有差异。也就是说，原假设可以用公式表示为：

133

$$H_0: P(\text{“破坏型错误”}) = P(\text{“非破坏型错误”}) = 1/2$$

设定 α 应该小于 0.10。实验者得到的数据如下：

- 有 11 个错误是“非破坏型错误”。
- 有 4 个错误是“破坏型错误”。

如果原假设为真，那么，当测试发现 15 个错误时“破坏型错误”不多于 4 个（即小于等于 4）的概率为：

$$P(0 \sim 4\text{个“破坏型错误”}) = \sum_{i=0}^{4}\binom{15}{i}\left(\frac{1}{2}\right)^{i}\left(\frac{1}{2}\right)^{15-i} = \frac{1}{2^{15}}\sum_{i=0}^{4}\binom{15}{i} = 0.059$$

也就是说，如果实验者根据得到的数据推断发现“非破坏型错误”比“破坏型错误”更普遍的结论，则其犯 I 类错误的概率为 0.059。在这种情况下，实验者可以拒绝 H_0，因为 $0.059 < 0.10$。

假如原假设为真，那么发现不多于 5 个“破坏型错误”的概率为 0.1509。这个值大于 0.10，意味着在 15 个错误中发现了 5 个“破坏型错误”时，不能拒绝 H_0。因此，当在实验中检测到的错误数为 15 时，实验者可以按照如下表述进行解释：

- 如果其中“破坏型错误”不超过 4 个，则拒绝 H_0。
- 如果其中“破坏型错误”超过 4 个，则无法拒绝 H_0。

总之，检测到的（15 个错误中）“破坏型错误”的数目是检验单元，而临界区域是 0，1，2，3，4（“破坏型错误”数）。

基于此，判定该检验的效能非常有意义。由于效能是当 H_0 不为真的时候拒绝 H_0 的概率，因此我们必须用公式明确地表达 H_0 不为真的含义。在本例中，“H_0 不为真”用公式可以表示为：

$$P(\text{“破坏型错误”}) < P(\text{“非破坏型错误”})$$

由于两个概率之和等于 1，它也能用如下公式描述：

$$P(\text{“破坏型错误”}) = a < 1/2$$

在 15 个错误中收到不多于 4 个“破坏型错误”的概率（也就是当 H_0 为假的时候拒绝 H_0 的概率）是：

$$p = \sum_{i=0}^{4}\binom{15}{i}a^{i}(1-a)^{15-i}$$

图 10-9 描绘了这个概率值随 a 值变化而变化的趋势。

由此可知，如果发现“破坏型错误”的概率和发现“非破坏型错误”的概率的差异大，则检验效能高。例如，当 $a = 0.05$ 时，发现“破坏型错误”的数目小于等于 4

的可能性很大。另一方面，如果差异很小，则检验效能会更小。例如，当 $a = 0.45$ 时，发现“破坏型错误”的数目大于 4 的可能性则会很大。 134

有一些因素会影响检验的效能。首先，检验本身效能会有差异；其次，样本的大小影响检验的效能。一个更大的样本意味着更高的效能。还有一个方面可能影响检验效能，即选择单边备择假设还是选择双边备择假设。单边假设比双边假设的检验效能更高。

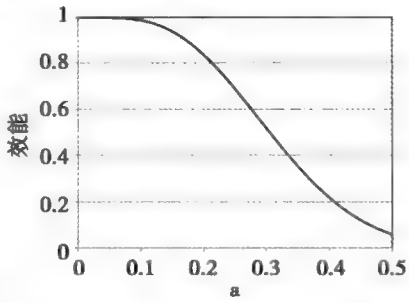


图 10-9　单边二项式检验的效能

Dybå 等人对假设检验在软件工程实验中的效能进行了评估，并且进行了深入的探讨［49］。

10.3.2　参数检验和非参数检验

检验可以分为参数检验和非参数检验。参数检验是建立在某种特定分布模型上的。在大多数情况下，我们会假设参数检验中涉及的某些参数是符合正态分布的。卡方检验是一种正态性检验，在下面讨论不同的检验类型时，将对其进行更进一步的介绍。参数检验还要求参数至少是通过定距尺度度量的，否则无法使用参数检验。此时，就需要使用各种非参数检验方法。

非参数检验关于参数分布的假设类型与参数检验的不同。在进行非参数检验时，我们只需要做些非常通用的假设。比如，之前小节中描述过的二项式检验就是一种非参数检验。非参数检验比参数检验更通用。这意味着如果样本可以使用参数检验方法来检验，通常只要有可用的非参数检验可用，都可以用非参数检验代替参数检验；但当使用非参数检验时，却往往无法使用参数检验来代替。关于选择参数检验还是非参数检验，需要考虑以下两个因素。 135

（1）适用性：不同的检验需要满足的假设分别是什么？确保关于参数分布的假设成立和假设所使用的尺度符合要求是非常重要的。

（2）效能：参数检验的检验效能通常高于非参数检验。因此，如果假设为真，则参数检验比非参数检验所需的数据点更少，实验也会更小。

Briand 等人也对参数和非参数统计方法的选择进行了讨论［27］。在他们的讨论中描述了在参数检验需要的条件无法满足的情况下，虽然使用参数检验方法存在一定的风险，但在有些情况下，仍然值得冒险尝试。仿真实验结果显示，只要偏差不是很大，参数检验方法（比如将要介绍的 t-检验）对于偏离前置条件（定距尺度）有着相当强的鲁棒性。

10.3.3　检验综述

除了上面介绍的二项式检验外，本节将介绍以下几种检验方法：

（1）t-检验（t-test）：最常使用的一种参数检验。该检验用于比较两个样本的均

值。这也意味着实验设计采用的是单因子双处置设计。

（2）Mann-Whitney 检验：一种用于代替 t-检验的非参数检验。

（3）F-检验（F-test）：一种用于比较两个样本分布情况的参数检验。

（4）配对 t-检验（Paired t-test）：一种针对成对比较设计的 t-检验。

（5）Wilcoxon 检验：一种用于代替配对 t-检验的非参数检验。

（6）符号检验（Sign test）：一种用于代替配对 t-检验的非参数检验。符号检验是 Wilcoxon 检验的一种更简单的替代处置。

（7）ANOVA 检验：一类用于单因子多级别设计的参数检验的统称。例如，ANOVA 检验能够用于单因子多级别设计、单因子与块变量设计、阶乘设计和嵌套设计等设计类型。

（8）Kruskal-Wallis 检验：一类在单因子多处置（大于 2）情况下代替方差分析的非参数检验。

（9）卡方检验（Chi-2）：一种针对频率数据的非参数检验。

根据设计类型和是否是参数或者非参数检验对不同的检验进行分类，结果如表 10-3所示。

表 10-3　不同设计类型对应的参数检验与非参数检验概览

设　计	参 数 检 验	非参数检验
单因子单处置		卡方检验、二项式检验
单因子双处置，完全随机设计	t-检验，F-检验	Mann-Whitney 检验、卡方检验
单因子双处置，成对比较	配对 t-检验	Wilcoxon 检验，符号检验
单因子多处置（大于 2）	方差分析	Kruskal-Wallis 检验，卡方检验
多因子	方差分析①	

①本书中没有描述这种检验。可以参阅 Marascuilo and Serlin [119] 和 Montgomery [125]。

对上述描述的所有检验，下文将按照以下维度用独立的表格依次进行介绍：

（1）输入：检验能够应用的度量类型。也就是说，输入描述了对实验设计的需求。只有满足需求的输入才能使用该检验方法进行检验。

（2）原假设：提供一个原假设的公式化描述。

（3）计算：基于度量数据描述应该如何计算。

（4）标准：拒绝原假设的标准。通常会涉及查统计表，在本书附录 B 中进行了描述。虽然本书仅提供了一种显著性水平下的对照表，但可根据书中给出的参考文献找到更加详尽的对照表。

这里，所有的检验描述都不完整。如果需要更多相关信息，可以参考正文中给出的参考文献。例如，Mann-Whitney 检验、Wilcoxon 检验、符号检验和 Kruskal-Wallis 检验，都只使用了少量样本用最简单的案例进行了介绍。如果样本很多（比如符号检验中样本数量超过 35），那么，在很多情况下是很难计算和做决策的（原因将在下面讨论）。在这种案例中，由于样本很多，可以根据样本进行某种近似估计。如何估计可参

见 Siegel 和 Castellan[157] 的论述，其中还描述了如何处理检验中发生等值（两个或者两个以上的相等值）的情况。

本文中对检验描述的目标是：使得读者能基于描述和示例使用该检验。因此，并不需要提供公式推导等细节信息。

使用上面介绍的描述方式，我们在 10.3.1 节中介绍的检验示例（二项式检验）可总结为表 10-4。

表 10-4　二项式检验

项　目	描　述
输入	两种不同类型的事件（事件 1 和事件 2）发生的次数
H_0	P（事件 1）$=P$（事件 2）
计算	计算 $p=\frac{1}{2^N}\sum_{i=0}^{n}\binom{N}{i}$，其中 N 是事件发生的总次数，n 是稀有事件发生的次数
标准	双边检验备择假设（H_1：P(事件 1)$\neq P$(事件 2)）：若 $p<\alpha/2$，则拒绝 H_0 单边检验备择假设（H_1：P(事件 1)$<P$(事件 2)）：若 $p<\alpha$，则拒绝 H_0，并且事件 1 是样本中的稀有事件

在上述表格中，二项式检验的原假设被描述为两个事件发生的可能性是相等的。当然，也可以用其他方式建立原假设，比如声明 P(事件 1)$=0.3$，P(事件 2)$=0.7$。至于如何在案例中执行检验，可以参考 Siegel 和 Castellan[157] 等文献。

对于本章中介绍的大部分检验，我们都举例说明了其如何使用。这些示例中使用的数据都是虚构的。此外，这些检验的显著性水平主要设定为 5%，在附录 B 中提供了可供查询的对照表。

更多详尽的对照表可以在 Marascuilo 和 Serlin[119] 以及 Montgomery[125] 等与统计学相关的书籍中查到。

10.3.4　t-检验

t-检验是一种用于比较两个独立样本的参数检验。也就是说，设计应该是单因子双处置的。t-检验能够基于许多不同的假设执行，但这里仅描述一种经常用到的处置。如需更多信息，可以参考 Montgomery[125]、Siegel 和 Castellan[157]，以及 Marascuilo 和 Serlin[119] 的示例。检验按照表 10-5 执行。

表 10-5　t-检验

项　目	描　述
输入	两个独立样本：x_1，x_2，…，x_n 和 y_1，y_2，…，y_m
H_0	$\mu_x=\mu_y$，即两个样本的期望均值是相等的
计算	计算 $t_0=\frac{\bar{x}-\bar{y}}{S_p\sqrt{\frac{1}{n}+\frac{1}{m}}}$，其中 $S_p=\sqrt{\frac{(n-1)S_x^2+(m-1)S_y^2}{n+m-2}}$，并且 S_x^2 和 S_y^2 是独立样本方差

（续）

项　目	描　述
标准	双边检验备择假设（H_1：$\mu_x \neq \mu_y$）：若 $\lvert t_0 \rvert > t_{\alpha/2,n+m-2}$，则拒绝 H_0。这里，$t_{\alpha,f}$是自由度 f 等于 $n+m-2$ 的 t 分布在显著性水平 α 上的临界值。分布可参见附表 B-1 和文献 Montgomery[125] 以及 Marascuilo 和 Serlin[119]。 单边检验备择假设（H_1：$\mu_x > \mu_y$）：若 $t_0 > t_{\alpha,n+m-2}$，则拒绝 H_0

t-检验的示例。比较两个项目中不同程序的缺陷密度。其中一个项目的结果是：

$$x = 3.42, 2.71, 2.84, 1.85, 3.22, 3.48, 2.68, 4.30, 2.49, 1.54$$

另一个项目的结果是：

$$y = 3.44, 4.97, 4.76, 4.96, 4.10, 3.05, 4.09, 3.69, 4.21, 4.40, 3.49$$

原假设为两个项目的缺陷密度相同，备择假设为两个项目中缺陷密度不同。由数据可知，$n=10$，$m=11$。x、y 的均值分别为 $\bar{x} = 2.853$ 和 $\bar{y} = 4.1055$。

由计算可以得到：$S_x^2 = 0.6506$，$S_y^2 = 0.4112$，$S_p = 0.7243$，$t_0 = -3.96$。

自由度 $f = n + m - 2 = 10 + 11 - 2 = 19$。通过查表 B-1 可知 $t_{0.025,19} = 2.093$。由于 $\lvert t_0 \rvert > t_{0.025,19}$，可知如果使用双侧检验，则在显著性水平 0.05 下可以拒绝原假设。

10.3.5　Mann-Whitney 检验

Mann-Whitney 检验是一种替代 t-检验的非参数检验。当不能确定样本数据是否能够满足 t-检验的假设要求时，可以使用该检验来替代 t-检验。Mann-Whitney 检验是一种基于序的检验方法，这里我们没有完整地描述它。更多细节可以参见 Siegel 和 Castellan[157]⊖ 以及 Marascuilo 和 Serlin[119] 等文献。表 10-6 给出了该检验的执行步骤。

表 10-6　Mann-Whitney 检验

项　目	描　述
输入	两个独立样本：x_1，x_2，…，x_n 和 y_1，y_2，…，y_m
H_0	两个样本的分布相同
计算	对所有样本进行排序并计算 $U = N_A N_B + \dfrac{N_A(N_A+1)}{2} - T$ 和 $U' = N_A N_B - U$，其中 $N_A = \min(n, m)$，$N_B = \max(n, m)$，T 是小样本的秩和
标准	根据计算结果查找拒绝原假设的临界值表进行判定。临界值表可参见表 B-3 或 Marascuilo 和 Serlin[119] 如果 $\min(U, U')$ 小于等于 B-3 中的值，则拒绝 H_0

Mann-Whitney 检验的示例。使用上述 t-检验示例中的数据（将所有数据按从小到

⊖ Siegel 和 Cestellan[157] 中描述了 Wilcoxon-Mann-Whitney 检验而不是 Mann-Whitney 检验。但这两个检验本质上是相同的。

大顺序排列，再使用序号代替原有数据），可以得到 $N_A = \min(10,11) = 10$，$N_B = \max(10,11) = 11$。小样本（x）所对应的秩为 9、5、6、2、8、11、4、17、3、1，大样本（y）所对应的秩为 10、21、19、20、15、7、14、13、16、18、12。根据秩可以计算得出 $T = 66$，$U = 99$，$U' = 11$。由于 U 和 U' 中的最小值小于 26，由表 B-3 可知，对于显著性水平为 0.05 的双侧检验而言，可以拒绝原假设。

136
139

10.3.6　F 检验

F 检验是一种用于比较两个独立样本的方差的参数检验。关于 F 检验，详见 Montgomery[125]、Robson[144]、Marascuilo 和 Serlin[119] 等文献。表 10-7 给出了 F 检验的执行步骤。

表 10-7　F 检验

项　目	描　述
输入	两个独立样本：x_1，x_2，…，x_n 和 y_1，y_2，…，y_m
H_0	$\sigma_x^2 = \sigma_y^2$，即两样本的方差相等
计算	计算 $F_0 = \dfrac{\max(S_x^2, S_y^2)}{\min(S_x^2, S_y^2)}$，其中 S_x^2 和 S_y^2 是两个独立样本的方差
标准	双边检验备择假设（H_1：$\sigma_x^2 \neq \sigma_y^2$）：若 $F_0 > F_{\alpha/2, n_{\max}-1, n_{\min}-1}$，则拒绝 H_0。其中 $n_{\max}$ 指方差大的样本的数据个数，$n_{\min}$ 指方差小的样本的数据个数。$F_{\alpha/2, f_1, f_2}$ 是自由度为 f_1 和 f_2、显著性水平为 α 的 F 分布的临界值，可以通过查表 B-5 或文献 Montgomery[125] 与 Marascuilo 和 Serlin[119] 等获得 单边检验备择假设（H_1：$\sigma_x^2 > \sigma_y^2$）：若 $F_0 > F_{\alpha, n_{\max}-1, n_{\min}-1}$ 并且 $S_x^2 > S_y^2$，则拒绝 H_0

F 检验的示例。同样使用上述 t-检验中示例的数据，可知两组数据的方差分别为 $S_x = 0.6506$，$S_y = 0.4112$，即由计算得 $F_0 = 1.58$，$n_{\max} = 10$，$n_{\min} = 11$。

由查表 B-5 可知，$F_{0.025,9,10} = 3.78$。由于 $F_0 < F_{0.025,9,10}$，因此使用显著性水平为 0.05 的双侧检验无法拒绝原假设。也就是说，该检验无法拒绝“这两个样本具有相同的方差”的原假设。

10.3.7　配对 t-检验

配对 t-检验主要用于比较从重复度量中获得的两个样本。这意味着在实验中对于一个实验主体进行了多次度量。例如，假如需要比较两种工具的性能，如果使用两个组独立地使用这两种不同的工具，那么产生的结果将会是两个独立样本，从而可以使用普通 t-检验进行检验。相反，如果实验主体仅为一个组，同时要求组内的每一个人都要使用这两种工具进行实验，那么，这就意味着需要重复度量。此时，可以利用配对 t-检验检验每个人使用不同工具的性能差异。

Montgomery[125]、Marascuilo 和 Serlin[119] 等对本检验进行了更加详尽的描述。检验的执行步骤参见表 10-8。

表 10-8　配对 t-检验

项　目	描　述
输出	成对样本：$(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$
H_0	$\mu_d = 0$，其中 $d_i = x_i - y_i$，即成对样本间差异的期望均值为 0
计算	计算 $t_0 = \dfrac{\bar{d}}{S_d/(\sqrt{n})}$，其中 $S_d = \sqrt{\dfrac{\sum_{i=1}^{n}(d_i - \bar{d})^2}{n-1}}$
标准	双边检验备择假设（H_1：$\mu_d \neq 0$）：若 $\lvert t_0 \rvert > t_{\alpha/2, n-1}$，则拒绝 H_0。这里，$t_{\alpha,f}$是自由度为 f、显著性水平为 α 的 t 分布的临界值。该临界值可以通过查找表 B-1 或文献 Montgomery[125] 以及 Marascuilo 和 Serlin[119] 等得到 单边检验备择假设（H_1：$\mu_d > 0$）：若 $\lvert t_0 \rvert > t_{\alpha, n-1}$，则拒绝 H_0

配对 t-检验的示例。十个程序员独立地开发了两个不同的程序。他们记录了开发程序所需要的工作量，如表 10-9 所示。

表 10-9　所需的工作量

程　序　员	1	2	3	4	5	6	7	8	9	10
程序 1	105	137	124	111	151	150	168	159	104	102
程序 2	86.1	115	175	94.9	174	120	153	178	71.3	110

原假设是开发程序 1 所需的工作量和开发程序 2 所需的工作量是相同的。备择假设是两者工作量不一样。为了执行检验，需要进行以下计算：

$$d = \{18.9, 22, -51, 16.1, -23, 30, 15, -19, 32.7, -8\}$$

$$S_d = 27.358$$

$$t_0 = 0.39$$

自由度 $f = n - 1 = 10 - 1 = 9$。由查表 B-1 可知 $t_{0.025,9} = 2.262$。

由于 $t_0 < t_{0.025,9}$，因此显著性水平为 0.05 的双侧检验表明无法拒绝原假设。

10.3.8　Wilcoxon 检验

Wilcoxon 检验是一种可以替代配对 t-检验的非参数检验。使用这个检验的唯一需求是需要能够确定哪一对样本值间的差异是最大的，即能够对差异的大小进行排序。该检验是基于秩的，具体细节不在此讨论。如需了解详情，请参见 Siegel 和 Castellan[157] 以及 Marascuilo 和 Serlin[119] 等文献。表 10-10 对这种检验进行了总结。

表 10-10　Wilcoxon 检验

项　目	描　述
输入	成对样本 s：$(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$
H_0	在不考虑符号的情况下对所有偏差（$d_i = x_i - y_i$）进行排序（1，2，3，…），那么，所有正偏差的秩和等于所有负偏差的秩和

（续）

项　目	描　述
计算	计算所有正偏差的秩和 d_i：s，记为 T^+；计算所有负偏差的秩和 d_i：s，记为 T^-
标准	基于 T^+、T^- 以及数据对的数量 n，通过查表来判断是否拒绝 H_0。例如，根据对照表 B-4，若 min（T^+，T^-）小于等于表 B-4 中的值，则拒绝 H_0。也可参照文献 Siegel 和 Castellan[157] 以及 Marascuilo 和 Serlin[119] 中的对照表

Wilcoxon 检验的示例。使用上述配对 t-检验示例中的数据，可以得到差异（d）的绝对值的等级排列是 4，6，10，3，7，8，2，5，9，1。基于此，能够计算出 T^+ 和 T^- 分别是 32 和 23。

由于 T^+ 和 T^- 的最小值都大于 8（见表 B-4），因此，使用显著性水平为 0.05 的双侧检验无法拒绝原假设。

10.3.9　符号检验

符号检验和 Wilcoxon 检验一样，也是一种可用于替换配对 t-检验的非参数检验。由于符号检验只需要利用每个配对中值的差异的符号进行计算，因此当不可能或者不需要对差异大小进行排序时，可以使用符号检验来替代 Wilcoxon 检验。例如，当采用符号检验就能显示出显著性的时候，就没必要使用 Wilcoxon 检验了。这是因为符号检验效能更低，也更容易执行。

符号检验在 Siegel 和 Castellan[157] 以及 Robson[144] 等人的文章中有更为深入的描述。表 10-11 对符号检验进行了总结。

表 10-11　符号检验

项　目	描　述
输入	成对样本：$(x_1,y_1),(x_2,y_2),\cdots,(x_n,y_n)$
H_0	$P(+)=P(-)$，其中 + 和 - 分别表示事件 $x_i>y_i$ 和 $x_i<y_i$
计算	用一个“+”表示每一个正偏差（$d_i=x_i-y_i$），同时用一个“-”表示每一个负偏差。计算 $p=\frac{1}{2^N}\sum_{i=0}^{n}\binom{N}{i}$，其中 N 为符号总数，n 为数量最少的符号的数量
标准	双边检验备择假设（H_1：$P(+)\neq P(-)$）：若 $p<\alpha/2$，则拒绝 H_0 单边检验备择假设（H_1：$P(+)<P(-)$）：若 $p<\alpha$ 并且“+”事件是样本中少数事件，则拒绝 H_0

读者可能注意到符号检验是一种二项式检验，其中的事件有两种：“+”和“-”。

符号检验的示例。使用上述配对 t-检验示例中的数据，可知有 6 个正偏差和 4 个负偏差。这意味着：

$$p=\frac{1}{2^{10}}\sum_{i=0}^{4}\binom{10}{i}=\frac{193}{512}\approx 0.3770$$

140
~
142

由于 $p>0.025$，因此用显著性水平为 0.05 的双侧检验无法拒绝原假设。

10.3.10　方差分析

方差分析（ANalysis Of VAriance，ANOVA）能够用于分析多种不同设计的实验。使用“方差分析”这个名字是因为它关注于数据的总体变化以及不同类别的各个分组数据的变化。例如，可以通过 ANOVA 检验“因处置不同产生的变化”与“因随机误差产生的变化”之间是否具有差异。

本节描述了在最简情形下如何使用方差分析，即用方差分析比较一些样本是否具有相同的均值。这也意味着，实验采用单因子多处置（大于 2）设计。表 10-12 对 ANOVA 进行了总结。

表 10-12　针对单因子多处置（大于 2）设计的方差分析

项　目	描　述
输入	a 个样本：x_{11}，x_{12}，…，x_{1n_1}；x_{21}，x_{22}，…，x_{2n_2}；…；x_{a1}，x_{a2}，…，x_{an_a}
H_0	$\mu_{x_1}=\mu_{x_2}=\cdots=\mu_{x_a}$，也就是说，所有期望的均值是相等的
计算	$SS_T=\sum_{i=1}^{a}\sum_{j=1}^{n_i}x_{ij}^2-\frac{x_{..}^2}{N}$ $SS_{Treatment}=\sum_{i=1}^{a}\frac{x_{i.}^2}{n_i}-\frac{x_{..}^2}{N}$ $SS_{Error}=SS_T-SS_{Treatment}$ $MS_{Treatment}=SS_{Treatment}/(a-1)$ $MS_{Error}=SS_{Error}/(N-a)$ $F_0=MS_{Treatment}/MS_{Error}$ 其中 N 是所有样本数据的总数，一个点表示遍历该点所代表的所有索引求和，例如，$x_{i.}=\sum_j x_{ij}$
标准	如果 $F_0>F_{\alpha,a-1,N-a}$，则拒绝 H_0。这里，F_{α,f_1,f_2} 表示自由度为 f_1 和 f_2、显著性水平为 α 的 F 分布的临界值。可以通过查表 B-5 和文献 Montgomery[125] 以及 Marascuilo 和 Serlin[119] 等来获得临界值

ANOVA 检验的结果往往通过方差分析表展示。例如，一个单因子多处置的 ANOVA 检验的结果如表 10-13 所示。

表 10-13　表 10-12 描述的 ANOVA 检验的方差分析表

变化来源	平方和	自由度	均方差	F_0
处置间	$SS_{Treatment}$	$a-1$	$MS_{Treatment}$	$F_0=\frac{MS_{Treatment}}{MS_{Error}}$
误差[1]	SS_{Error}	$N-a$	MS_{Error}	
总计	SS_T	$N-1$		

[1] 有时标记为“处置内”。

需要注意的是，上述 ANOVA 检验仅仅是 ANOVA 检验的一种变体。方差分析能够用于不同因素、不同分块变量的多种不同的设计。限于篇幅，不在此对这些检验方式进行详细描述。更多详细信息可以参阅 Montgomery[125]、Marascuilo 和 Serling[119]等文献。 143

方差分析的示例。对三个不同程序的模块大小进行测量后，得到如下数据：

程序 1：221,159,191,194,156,238,220,197,197,194

程序 2：173,171,168,286,206,140,226,248,189,208,213

程序 3：234,188,181,207,266,153,190,195,181,238,191,260

原假设是“三个程序中模块大小的均值相等”，备择假设是其均值不相等。基于上述数据，计算得到如表 10-14 所示的方差分析表。

表 10-14　方差分析表

变化来源	平方和	自由度	均方差	F_0
处置间	579.0515	2	289.5258	0.24
误差	36 151	30	1 205	
总计	36 730	32		

自由度 $f_1 = a - 1 = 3 - 1 = 2$，$f_2 = N - a = 33 - 3 = 30$，查表 B-5 可得 $F_{0.025,2,30} = 4.18$。由于 $F_0 < F_{0.025,2,30}$，所以在显著性水平为 0.025 时不能拒绝原假设。

10.3.11　Kruskal-Wallis 检验

Kruskal-Wallis 检验是一种基于序的方差分析方法。它是一种可以用于替代上述单因子方差分析的非参数检验方法。如果不能保证满足参数检验 ANOVA 的假设，则可以用这种方法来替代。本节不会具体描述这种基于序的检验方法，有兴趣请参见 Siegel 和 Castellan[157] 以及 Marascuilo 和 Serlin[119] 等文章。

表 10-15 对这种检验进行了总结。

表 10-15　Kruskal-Wallis 检验

项　目	描　述
输入	a 个样本：$x_{11}, x_{12}, \cdots, x_{1n_1}; x_{21}, x_{22}, \cdots, x_{2n_2}; \cdots; x_{a1}, x_{a2}, \cdots, x_{an_a}$
H_0	a 个样本的中位数相等
计算	对所有样本值进行排序（1，2，…，$n_1 + n_2 + \cdots + n_a$），根据该序，计算每个样本的中位数，参见文［119，157］中的示例
标准	参见诸如 Siegel 和 Castellan[157] 以及 Marascuilo 和 Serlin[119] 等文献

144

10.3.12　卡方检验

卡方检验（有时表示为 χ^2 检验）能够以各种不同方式执行。但所有的卡方检验检

验的数据都是频率数据。例如，有两个系统，每个系统均由很多模块构成。系统 1 有 15 个小模块、20 个中等模块和 25 个大模块；系统 2 有 10 个小模块、19 个中等模块和 28 个大模块，如表 10-16 所示。

表 10-16　两个系统（群组）的模块大小（变量）的频率表

模块大小	系　统　1	系　统　2
小	15	10
中	20	19
大	25	28

在本案例中，将通过卡方检验判断两个系统中小模块、中等模块和大模块的分布是否相同。

卡方检验也能够针对一组数据进行检验，判定该频率分布是否与理论分布相同。例如，利用卡方检验检查样本是否符合正态分布。

卡方检验用于判定两组或者多组的度量值是否来自于同一个分布，其总结如表 10-17 所示。

表 10-17　卡方检验，k 个独立样本（组）

<table>
<tr><th>项　目</th><th colspan="4">描　述</th></tr>
<tr><td>输入</td><td colspan="4">k 组频率数据</td></tr>
<tr><td>H_0</td><td colspan="4">k 组数据源于同一个分布</td></tr>
<tr><td rowspan="3">计算</td><td colspan="4">创建一个列联表。对于一个由包含三种变量取值的两组数据构成的示例（即与表 10-16 中的数据维度相同）创建如下所示的列联表：</td></tr>
<tr><td colspan="4"><table>
<tr><th>变量</th><th>组 1</th><th>组 2</th><th>合计</th></tr>
<tr><td>1</td><td>n_{11}</td><td>n_{12}</td><td>R_1</td></tr>
<tr><td>2</td><td>n_{21}</td><td>n_{22}</td><td>R_2</td></tr>
<tr><td>3</td><td>n_{31}</td><td>n_{32}</td><td>R_3</td></tr>
<tr><td>总计</td><td>C_1</td><td>C_2</td><td>N</td></tr>
</table></td></tr>
<tr><td colspan="4">其中，n_{ij}代表组 j 中变量取值为 i 时出现的频率，C_i 代表组 i 中所有变量取值出现的频率总数，R_i 代表变量取值为 i 时出现的频率总数。N 代表所有频率的总数
计算 $X^2=\sum_{i=1}^{r}\sum_{j=1}^{k}\frac{(n_{ij}-E_{ij})^2}{E_{ij}}$，其中 $E_{ij}=\frac{R_iC_j}{N}$（E_{ij}表示 H_0 为真时的期望频率），r 为变量数，k 为群组数</td></tr>
<tr><td>标准</td><td colspan="4">若 $X^2>\chi^2_{\alpha,f}$，则拒绝 H_0。其中自由度 $f=(r-1)(k-1)$，$\chi^2_{\alpha,f}$是自由度为 f、显著性水平为 α 的卡方分布临界值，可通过查表 B-2 或 Siegel 和 Castellan 的文献［157］获得</td></tr>
</table>

卡方检验的示例。如果使用卡方检验检验表 10-16 中的数据，可以构建表 10-18。

表 10-18　卡方检验的计算结果（期望值 E_{ij} 列在括号内）

模块大小	系　统　1	系　统　2	合　　计
小	15（12.8205）	10（12.1795）	$R_1=25$
中	20（20）	19（19）	$R_2=39$
大	25（27.1795）	28（25.8205）	$R_3=53$
总计	$C_1=60$	$C_2=57$	$N=117$

原假设是两个系统的模块大小的分布相同，备择假设是该分布不同。基于这些数据，我们可以计算出检验统计量为 $X^2=1.12$。自由度为 $(r-1)(k-1)=2\times1=2$。查表 B-2 可知，$\chi^2_{0.05,2}=5.99$。由于 $X^2<\chi^2_{0.05,2}$，因此，在显著性水平为 0.05 时无法拒绝原假设。

卡方拟合度检验。卡方检验也能用来检验度量值是否符合某特定分布，如正态分布。对本例而言，其拟合度检验步骤如表 10-19 所示。

表 10-19　卡方拟合度检验

项　　目	描　　述
输入	一组频率类数据（即 O_1，O_2，…，O_n，其中 o_i 代表在分类 i 中被观测对象的出现次数），与表 10-2 类似
H_0	度量值服从某特定分布
计算	计算 $X^2=\sum_{i=1}^{n}\frac{(O_i-E_i)^2}{E_i}$，其中 E_i 是在 H_0 为真时分类 i 中期望的观察次数，n 是分类的数量
标准	若 $X^2>\chi^2_{\alpha,f}$，则拒绝 H_0。其中自由度 $f=n-e-1$ 且 e 为根据源数据估计的参数个数。$\chi^2_{\alpha,f}$ 是自由度为 f、显著性水平为 α 的卡方分布的临界值。其值可通过查表 B-2 或文献 Siegel 和 Castellan[157] 获得。本检验是个单侧检验

如果需要针对一个连续分布进行拟合度检验，那么必须将数据划分到各个区间内，使得每个区间可以代表一个值。以正态分布为例，分析如下。

假如 H_0 的分布是已知的（例如，$P(X=1)=2/3$，$P(X=2)=1/3$），那么，就没有必要从测量数据中估计参数（此时 $e=0$）。另一方面，如果原假设仅仅声明其值服从正态分布，那么就必须估计两个参数——正态分布的均值和标准差，否则就不可能确定出不同区间的期望值 E_i。因此，在本例中，$e=2$。

*示例：正态分布的卡方拟合度检验。*60 个学生开发同样的程序，其开发的程序代码行数如表 10-20 所示。

表 10-20　程序规模

757	758	892	734	800	979	938	866	690	877	773	778
679	888	799	811	657	750	891	724	775	810	940	854

（续）

784	843	867	743	816	813	618	715	706	906	679	845
708	855	777	660	870	843	790	741	766	677	801	850
821	877	713	680	667	752	875	811	999	808	771	832

原假设是数据呈正态分布，备择假设是数据不服从正态分布。由数据分析可以计算出其均值和标准差分别为 $x=794.9833$，$s=83.9751$。

如果数据确实服从均值为 x、标准差为 s 的正态分布，那么，可以将分布的值域划分为若干段（子区域），使得每段包含的值的概率相同。在此例中，将整个值域划分为 10 段。为了找到第一个分段的上限（x），必须求解下面的等式：

$$P(X < x) = 1/10$$

其中 X 服从正态分布 $N(\bar{x}, s)$，根据标准正态分布定义可知，相当于求解：

$$P(X_s < (x-\bar{x})/s) = 1/10$$

其中 X_s 服从标准正态分布 $N(0, 1)$，这也等同于求解：

$$P(X_s < z) = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2/\pi}} e^{(-y^2)/2} dy = 1/10$$

其中 X_s 服从标准正态分布 $N(0, 1)$ 且 $x=sz+\bar{x}$。

这些等式能够使用多种不同方式求解。其中一种方法是迭代，利用计算机帮助求解 z 和 x；另一种方法是使用标准正态分布表（表中罗列了不同的 z 值所对应的 $p(X_s < z)$），这种表格在大多数统计学书籍中都有；也可以使用一种能够直接显示分段临界值（即 z 值）的专用表格，可参见 Humphrey[82]。

表 10-21 展示了分段临界值和实际落入每个分段的值的数量。

表 10-21　分段

分　段	下　界	上　界	值的数量
1		687.3	8
2	687.3	724.3	6
3	724.3	750.9	4
4	750.9	773.7	6
5	773.7	795	5
6	795	816.3	9
7	816.3	839	2
8	839	865.7	6
9	865.7	902.6	9
10	902.6		5

每个分段中数值个数的期望值 E_i 为 60/10 = 6。这意味着卡方值 $\chi^2 = 7.3$。自由度是 10 − 2 − 1 = 7。由表 B-2 可得 $\chi^2_{0.05,7} = 14.07$。由于 $X^2 < \chi^2_{0.05,2}$，所以在显著性水平为 0.05 时无法拒绝原假设。如果观察根据该数据所做的直方图（见图 10-10），不难看出其确实非常像正态分布。

145
~
147

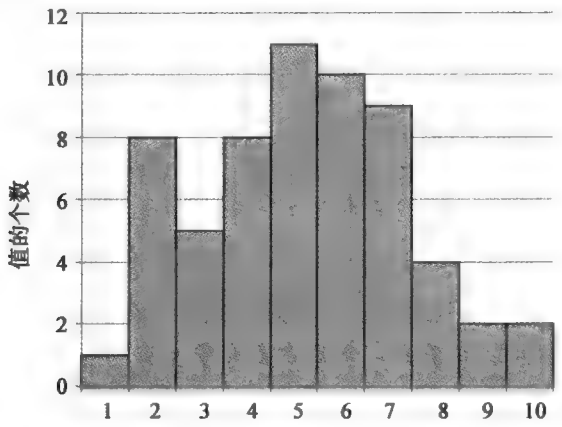


图 10-10　直方图

卡方检验的结束语。卡方检验是基于某种确定性假设而设计的，一般当期望值 E_i 不太小时可以使用。使用时必须遵循的经验法则是：如果自由度 f 等于 1 且有任何期望频率小于 5 时，不可使用卡方检验；如果 $f > 1$ 且超过 20% 的期望频率小于 5 或者有任何的期望频数小于 1，也不可以使用卡方检验。值得注意的是，有时候尽管期望频率不满足要求，也使用了本检验。但在这样的情况下，将存在计算风险。

一种获得更高的期望频率的方法是把相关的类别合并成新的类别。但合并得到的新类别必须是有意义的。更多关于卡方检验的信息可以参考 Siegel 和 Castellan[157] 的文章。

10.3.13　模型充分性检查

每个统计模型都依赖于某种特定的假设，例如分布、独立性和尺度等。如果数据集不能满足前提假设，那么假设检验的结果也是无效的。因此，检查所有的假设是否都能被满足是至关重要的。

模型充分性检查依赖于以下三个前提假设：

（1）正态性：如果某种检验需要数据符合正态分布，可以使用卡方检验评估数据能多大程度满足该假设。卡方检验已经在上文中描述过。

（2）独立性：如果检验要求数据来自于一个有若干独立随机变量的样本，那么，必须检查样本集之间不存在相关性。这可以用散点图和计算相关系数的方法来检查。相关方法在本节开始已有介绍。

（3）残差：在许多统计模型里，有一个术语表示残差（统计误差）。通常假设残

148

差服从正态分布。校验这种性质的通常做法是将残差绘制在散点图里查看数据是否不存在某种特定的趋势（分布看上去是随机的）。

10.3.14　推导结论

分析和解释实验数据之后，我们需要根据实验输出得出结论。如果假设被拒绝，在实验有效的前提下，我们可以得出关于独立变量如何影响非独立变量的结论，如第8章所述。

另一方面，如果实验不能拒绝原假设，就无法得出任何关于独立变量影响非独立变量的结论。此时，我们唯一能够给出的解释是在两个处置之间不存在统计意义上的显著差异。

如果能够得到具有统计意义的显著差异，我们就会希望得出关于独立变量和非独立变量之间关系的通用结论。在达成这个目标之前，我们需要考虑实验的外部有效性，如第8章所述。我们只能将结果推广到跟实验环境类似的环境中。

尽管实验结果可能在统计意义上是显著的，但并不意味着实验结果就一定具有实际意义。例如，证明了方法X比方法Y的成本效益高2%具有很高的统计显著性，但用方法X替换方法Y可能并不划算。也就是说，必须研究不同处置观察到的效果大小并且基于此来得出结论和提出建议。Kampenes[92]等人给出了关于不同效果大小概念的综述，并且写了一篇关于如何处理这个问题的系统综述。

反之亦然，尽管实验结果可能在统计意义上不显著或者只有很低的统计显著性，我们仍有可能从实验中获得有重要实际意义的结论。在某个显著性水平下无法拒绝原假设并不意味着原假设就是真的。可能在实验设计中存在一些问题，例如存在一些对有效性的真实威胁或者数据样本太少的情况。此外，根据情况和研究目的，我们可以由于实验结果具有很高的现实意义而设定一个更低的统计显著性水平。这个问题与8.9节中关于有效性威胁的讨论是相关的。

当发现变量A和变量B之间是显著相关的时候，通常我们也不能得出A和B之间存在着因果关系的结论。因为可能存在着第三个因素C，造成了A和B的关联。

基于实验输出得出的结论是一个决策的输入，比如，将一种新方法应用于今后的
149 项目或者还需要进一步的实验来验证该方法。

需要注意的是使用假设检验也有一些弊端。如Miller[122]指出的那样，如果能够提供足够多的数据，大多数情况下都可以找到一种能够被拒绝的原假设的表述方式。但实际上我们很难获得一个能代表总体的样本，例如世界上所有的软件工程师。应当谨慎地基于实验结果采取某种措施，并且应该把实验结果仅看作决策过程的一个因素。

10.4　示例分析

本示例是9.4节中示例的延续。基于实验获得的数据，首先应用描述性统计分析数据，如绘制数据分析图表。采用不同尺度获得的数据应该使用不同的统计方法分析，

详见 10.1 节。其中一种常用的数据图表绘制方法是箱形图。箱形图能清晰地展示数据的概况、确定离群点。如果识别出一个离群点，了解是否有导致其出现的底层原因非常重要。例如，可能有一个或者少数实验主体和其他实验主体的背景不同，因此，必须确保他们的数据与其他主体的数据是可比的。如果只有一组数据受到影响，这就更加重要。一般而言，我们应该谨慎对待数据点的移除，即任何的数据移除都应该是目标明确和记录在案的。

一旦确定了在数据分析中应该包括哪些数据，就应该考虑统计分析了。关于不同统计方法的使用条件、什么时候应该用参数检验、什么时候应该用非参数检验，人们总是有许多不同的观点，因此，统计分析方法的选择常常充满挑战。

首先应该检查数据是否服从正态分布，例如，通过绘制直方图（见图 10-5）或者使用 10.3.12 节中描述的卡方检验，或者使用其他候选检验方法进行判定，比如 Kolmogorov-Smirnov 检验、Shapiro-Wilks 的 W 检验、Anderson-Darling 检验。然而，在样本量很小时，很可能数据本来不服从正态分布，但看起来却是呈正态分布的，并且正态性检验可能由于数据点太少而并没有检测出来。此时，某些参数检验比其他检验方法鲁棒性更强。例如，t-检验对于非正态性相当鲁棒，而方差分析则不行。简而言之，最好在分析前调查数据是否符合正态分布。

对双因素（阅读技术和需求文档）、每个因素双处置的设计而言，尤其需要数据服从正态分布。当数据服从正态分布时才能使用方差分析来检验。如果数据不服从正态分布，那么就会面临一个问题：没有非参数检验方法能够分析这种设计类型，如表 10-3所示。如果只是单因子双处置设计，那么，可以使用非参数检验方法替代。因此，即使有一些更简单的设计看起来非常适合、用起来也非常方便，但实际上使用这种类型的设计可能并不是一个好的选择。因为使用该设计虽然获得了更多的数据点，但可能会给统计分析带来挑战。因此，当我们选择实验设计时，尤其要考虑后果。这种设计类型有时候会涉及交叉设计，例如，主体先使用或知晓了一个处置，然后再让他们使用或知晓第二个处置。此时需要对一些挑战进行探讨，比如是否能够只考虑一个因素（如阅读技术）。然而，在两次审查时使用同一个需求文档是不现实的，除非两次审查之间间隔了一段很长的时间。Kitchenham 等人［99］提出了一些针对交叉设计的统计方面的挑战。文中指出，尽管其他人不推荐在软件工程领域使用交叉设计，但如果不使用交叉设计，就可能导致实践每个处置的主体太少，因此，在软件工程领域，权衡这两种处置的利弊，往往也会选择使用交叉设计［99］。 150

如果假设数据服从正态分布，那么可以使用方差分析来检验。但此时仍然存在一个挑战：即使方差分析得出了显著的结果，我们仍然不知道哪种差异是显著的。为了解决这个问题，必须在方差分析后进行一些其他检验，例如 Fisher 提出的 PLSD（Protected Least Significant Difference）检验［125］。PLSD 检验要求数据必须通过方差分析得到具有显著性差异的结果，即它受显著的方差分析结果的保护。Fisher 的 PLSD 检验可用于均值的成对比较。同样，它也会面临实验设计所带来一些统计意义上的挑战。

因此，需要使实验设计尽量简单，从而能够进行正确的统计分析。

如果我们选择将实验主体分成两组，要求他们对同一份需求文档应用 PBR 或 CBR 方法进行审查，那么，此种设计即为单因子双处置设计。这意味着接下来可以根据正态性检验的结果来决定使用 t-检验或是 Mann-Whitney 检验。但另一方面，这种实验设计使得我们无法分析出实验主体和处置之间的相互作用。因此，这种实验设计的优劣取决于个案的实验目的，同时也依赖于实验主体的数目和识别的有效性威胁。

10.5　练习

10.1　什么是描述性统计？描述性统计有什么用途？

10.2　分别解释什么是参数检验和非参数检验，它们能在什么时候使用？

10.3　什么是检验的效能？

10.4　什么是成对比较？

151　10.5　简要解释 ANOVA 检验。

归档与展示

当一个实验完成后，其结果可能会按照图 11-1 中定义的那样提交给不同的受众。通常有以下几种做法：在会议或者期刊上发表一篇文章，或为决策者提供一份报告，也可以打包为重复实验服务或作为教学材料。打包也可以在公司内部进行，以改进和理解不同的过程。在本案例中，根据文献［16］中 Basili 等人提出的概念，最好能将实验经验记录到经验库中。但本书主要关注用会议或期刊中学术报告的形式来记录。如果由于篇幅限制在会议或期刊中无法报告实验的所有细节，那么，建议同时发布一个完整的技术报告。

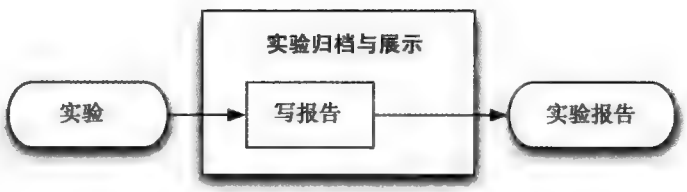


图 11-1　归档与展示过程概览

Jedlitschka 和 Pfahl 提出了一种撰写实验学术报告的模式［86］，Kitchenham 等人对其进行了评价［101］。表 11-1 总结了 Jedlitschka 和 Pfahl 的提案，11.1 节对其进行了简要的介绍。

表 11-1　Jedlitschka 和 Pfahl 提出的实验报告结构［86］

章节/子章节	内　容
标题、作者、结构化摘要	根据背景或情境、目标（或目的）、方法、结果、结论总结全文
动机	确定工作范围，鼓励读者读下去
问题陈述	报告问题是什么，在哪里发生，谁进行观察
研究目标	采用 GQM 中的规范样式定义实验
情境	报告环境要素，例如设置和位置
相关工作	当前研究与其他研究间的关系如何
实验设计	描述实验计划阶段的输出
目标、假设和变量	描述精化的研究目标
设计	定义实验设计的类型
实验主体	定义实验主体采样和分组所用到的方法
实验对象	定义用到的实验对象
操作指南	定义使用的任何操作指南或度量指南

（续）

章节/子章节	内　容
数据收集程序	定义实验计划、时机与数据收集程序
分析过程	详细说明使用的数学分析模型
有效性评估	描述材料、过程的有效性，确保参与者按照实验方法操作，用方法确保数据收集方法和工具的有效性和可靠性
执行	描述实验计划是如何实施的
样本	描述样本特征
准备	实验小组如何组成及培训
数据收集的执行	数据收集如何进行，与计划有何偏差
有效过程	如何按照有效过程实施，与计划有何偏差
分析	总结分析收集的数据，描述分析是如何进行的
描述性统计	采用描述性统计描述数据
数据集约简	描述对数据集的任何约简，如离群点的去除
假设检验	描述如何对数据进行评估和如何确认分析模型
解释	解释分析阶段的发现
结果评估和启示	解释结果
研究局限性	讨论对有效性的威胁
推论	针对得到的发现和局限性如何泛化结果
获得的经验教训	描述实验过程中做得好与做得不好的方面
结论与未来工作展望	描述研究的总结
与现有研究的关系	与先前实验对比，描述研究的贡献
影响	识别出最重要的发现
局限性	识别出方法的主要局限性，例如，无法取得预期结果的情形
未来工作展望	对进一步深入研究准备开展的其他实验的建议
致谢	列出不是作者的所有贡献者
参考文献	列出所有引用的文献
附件	包括原始数据或可能帮助他人使用结果的详细分析

11.1　实验报告的结构

结构化摘要。摘要呈现给读者的，应该是对实验关键特征进行的简要总结。实践证明，结构化摘要不仅是一种辅助数据抽取［30］的有效工具，同时也是写好摘要的有效工具。一个结构化摘要包含如下元素：

- 背景或者情境；
- 目标或目的；
- 方法；
- 结果；
- 结论。

示例：通过一个结构化摘要的实例来说明上述摘要中的五个元素。这里，结构化

摘要的长度被限制在 300 字以内。

情境：在市场营销领域，已有研究表明，在一个组织内，由于人们的职责和分工不同，当谈及公司内需要改进的事项时，不同角色对改进事项的优先级认识也不同。但在软件改进方面也是如此吗？

目标：本文评估了在一个软件开发组织中，不同角色如何看待软件过程改进中的不同问题以及这种差异能否为组织提供更多的个性化过程改进；同时，将此作为工作假设开展研究。

方法：开发了一个定量的调查问卷，包括五个不同权重的、与软件过程改进相关的问题。通过向瑞典一家电信公司全部 84 个雇员发放问卷开展调研，回收问卷 63 份。

结果：不同角色对其中 3 个问题有不同意见，而对另外两个问题意见一致。在改进的重要性、问题的紧急性和对成功过程管理的威胁三个方面的问题大家意见不同；而意见一致的问题则主要聚焦在过程间的交流（文档和教学）。

结论：总结认为，明确并考虑不同角色的不同需求是很重要的。这将使得为特定角色提供个性化改进成为可能，从而帮助他们克服对过程改进的抵触情绪。对于可以通过过程改进受益的其他领域和公司（如市场营销），本研究也很重要。

动机。动机或者介绍部分确定了研究范围、定义了研究目标。因此，在动机部分主要介绍确定研究范围阶段的结果（详见第 7 章）。介绍工作意图也能激发读者的兴趣。动机的介绍能让读者理解为什么做这项研究以及为什么需要做这项研究。同时，在这部分也应该简要介绍一下实验的情境。

相关工作。相关工作对于理解当前实验与以前所做的工作间的关系非常重要。虽然实验报告不需要对以前的工作做一个完整的系统文献综述（详见第 4 章），但系统地对已有的文献进行搜索比对仍然非常有益。特别是在重复研究中，前期研究都应该被说明。

实验设计。这里，应该介绍计划阶段的输出，详见第 8 章。由问题派生的假设应该在此进行详细描述。实验设计的描述中还应该包括设计类型、度量的变量（包括独立变量和非独立变量）以及实验操作指南。

153
~
155

实验设计中还应该包括关于如何收集与分析数据的描述以及对主体特征的描述。在此，也应该对实验的结论有效性、内部有效性、结构有效性和外部有效性进行讨论，同时对计划面临的威胁进行讨论。

描述这些方面的目的不仅是让其他人能够理解实验设计，从而使得读者相信结果是可信的，而且能够使研究可以被重复。总之，实验设计应该帮助读者更深入地理解实验。

执行。执行部分首先描述实验准备是如何做的，详见第 9 章。全方位的描述很重要，这能够使得重复实验易于执行，也能够使读者洞察这些活动是如何执行的。描述中必须包括实验主体的准备，如他们是否参加了某些课程的培训。实验的执行和实验期间的数据是如何收集的也应该被描述。

数据收集的确认过程是另一个必须强调的方面。如果有没有按照计划执行的步骤也一定要说明。描述这些信息的目的是提供一个数据有效的案例并突显存在的问题。

分析。数据分析部分应该描述使用某种特定分析模型的假设及其计算过程。描述应该包括样本大小、显著性水平和检验的应用等信息，以便读者了解分析的预备知识。在解释分析结果时应该阐明操作的原因，比如移除离群点的原因，从而避免误解。更多信息详见第 10 章。

解释。从分析得来的原始结果不足以理解实验结果和实验结论，必须提供解释，详见第 10 章，包括拒绝或者不能拒绝原假设的解释，以及总结应该如何使用从实验中得到的结果。

解释应与有效性相关，见第 8 章。对结果产生影响的因素都要予以描述。

结论和未来工作展望。最后，在结论部分应该对实验的发现和结论进行讨论，作为对整个实验的总结，包括实验的最终结果、问题以及与实验计划有出入的方面等。
156 实验结果也应该与先前介绍的相关工作关联起来。比较这些发现的异同点也很重要。

对未来工作的展望可以包括诸如以下方面的论述，如哪里能够找到更多信息以帮助读者更深入地理解实验、哪里能够找到更多信息以使重复实验更容易等。

附录。不重要的信息可以通过附录呈现。这些信息可以是收集到的数据、关于实验主体和客体的更多信息等。如果文章的意图是制作一个实验包，还可以在这里提供实验中用到的材料。

11.2　练习

11.1　为什么完整地记录实验非常重要？

11.2　什么是实验包？你在网上能找到实验包吗？

11.3　为什么介绍相关工作非常重要？

11.4　为什么仅仅提供分析的结果是不够的？换句话说，为什么对结果进行特定的解释至关重要？

157 11.5　做系统文献综述时哪类信息最重要？什么时候应该进行重复实验？

第三部分

Experimentation in Software Engineering

实验示例

第12章
Experimentation in Software Engineering

实验过程说明

本章的主要目的是通过举例说明前面章节中介绍的实验和实验过程。同时，本章的重点主要聚焦于实验过程而不是遵循第11章中提出的实验报告结构。

本章中给出的实验的目的是调查具有不同背景的人在参与课程《个体软件过程(Personal Software Process，PSP)》培训后使用PSP的表现是否存在差异。该实验是一个关于PSP对不同个体的表现差异的大型调查研究中的一部分。由于“个体背景”无法被随机地指派到实验主体上，因此该实验实际上是一个准实验。

PSP是一种用于管理和改进个人软件开发方式的自我持续改进过程，是一种支持软件开发的系统化方法。其过程包括：度量、评判、规划和追踪四个阶段。其中，重用是一个关键问题，特别是个体经验和数据的重用非常关键。PSP课程通过七个增量步骤介绍整个过程，并通过使用模板、表格和过程脚本为过程增加新的特性。

为了简单起见，我们在此只评估两个假设。此研究所用的数据集可以在附录A.1.2中找到。本章所讲述的实验采用的数据集是附录A.1.2的一个子集。

12.1 确定范围

12.1.1 目标定义

第一步要确定的是：实验是不是分析当前问题的一个恰当的方法。在本章的案例中，经验型研究的目的是在给定个体背景的前提下，判定其使用PSP方法的表现是否存在个体差异。

实验的动机在于，需要了解使用PSP方法时个体表现上的差异。众所周知，软件工程师的表现是千差万别的。引入个体软件过程的目的之一就是为个人提升其工作能力提供支持。为了尽可能支持这种改进，了解在使用PSP时可能有哪些预期差异、是否有可能解释这些差异，并进而理解个体差异是非常重要的。

研究对象 研究对象是PSP课程的学员以及这些学员基于其背景和经验所表现出的能力。Humphrey在他的两本书[82，83]中从研究对象的角度给出了个体软件过程PSP的定义。

目的 实验目的是基于参加PSP课程的个体背景评估个体的表现。通过实验洞察使用PSP方法与个体表现间的关系。

视角 本研究是从研究者和教师的视角出发开展研究。例如，研究者或者教师希望知道：参加PSP课程的学员的不同背景是否会导致其在课程中的表现出现系统性差异。那些希望将来参加PSP课程学习或者希望将PSP方法引入产业界的人士也同样希

望得知结论。

质量焦点　实验研究的主要效果是个体在 PSP 课程中的表现。其中，有两个度量指标：生产率（千行/开发时间）和错误率（错误数/千行），其中千行（KLOC）代表千行代码。

情境　本实验是在瑞典 Lund 大学通信系统系 1996～1997 学年 PSP 课程中进行的。与 Humhrey[82] 中阐述的 PSP 的主要不同之处在于，本实验采用了编码规范和行计数标准；同时，不论参与 PSP 课程学员的背景如何，本课程统一使用 C 语言作为指定编程语言。实验情境特征属于“单对象多检验研究”，如表 7-1 所示。本研究聚焦于 PSP，或者更确切地说，聚焦于 Humhrey[82] 中标记为 1A-10A 的 10 个程序。有许多学生学习了本课程（本年度共有 65 个学生完成了本课程的学习）。因此，从 7.1 节可以看出，本研究包括 65 个主体。因此，根据定义，本研究可以被判定为一个对照实验。学生的来源缺乏随机性，学生是选修 PSP 课程的学生，意味着本研究还缺乏一个使之成为一个完全的对照实验的重要因素。因此，本实验是一个准实验，参见 7.1 节。 162

12.1.2　范围总结

根据 7.2 节的要求，总结如下：

研究对象：PSP 的输出结果

目的：评价

质量焦点：个体的背景知识

视角：研究人员和教师的角度解释

情境：PSP 课程

12.2　计划

12.2.1　情境选择

本实验的情境为大学中的 PSP 课程，因此是在线下进行的（即不是在一个产业界的软件开发过程中进行的）。实验由毕业生完成（通常是大学四年级学生）；同时，由于它聚焦于教育环境下的 PSP，因此，实验具有特殊性。将本实验这种特定的情境中得出的结论泛化到其他情境的能力，将在讨论对实验的有效性威胁时进一步阐述。本实验讨论的是一个真实存在的问题：在 PSP 中个体表现的差异以及对这些差异的理解。

由于本实验定义清晰，所以将 PSP 的使用作为实验的情境为其他研究人员重现实验提供了绝佳的机会。同时，这也意味着没有必要在进行实验时花费太多的精力去定义和创建实验。Humhrey[82] 已经定义了实验情境，因此，在此没有必要为数据收集等活动准备表单。

12.2.2　构建假设

实验的一个重要方面就是明确并正式、清晰地声明实验将要评估什么。这将形成

一个或多个假设。这里，我们选择聚焦于以下两个假设：

1. 来自于计算机科学与工程（Computer Science and Engineering，CSE）培养计划和电气工程（Electrical Engineering，EE）培养计划的学生学习了 PSP 课程。通常情况
163 下，CSE 的学生学习过更多计算机科学与软件工程方面的课程，因此，预期他们比 EE 的学生生产率更高。

2. 作为课程第一讲的一部分，要求学生填写一张问卷调查表，调查他们与本课程相关的背景经验（如表 A-1 所示），例如，是否具备 C 语言的知识。无论学生先前是否拥有 C 语言的使用经验，都要求在课程中使用 C 语言。因此，这意味着不要求学生在参加 PSP 课程之前已经学习过 C 语言，同时也意味着部分学生需要在 PSP 课程上学习 C 语言。这不是根据 Humhrey［82］的推荐来做的。基于 C 语言经验的假设为，拥有更多 C 语言编程经验的学生在单位代码行中所犯的错误会更少。

基于上述假设的非正式陈述，我们可以对其进行规范描述，同时定义评估此假设所需的度量指标：

1. 原假设 H_0：CSE 的学生与 EE 的学生在生产率（利用单位开发时间开发的代码行数进行度量）方面没有差异。

原假设 H_0：Prod(CSE) = Prod(EE)

备择假设 H_1：Prod(CSE)≠Prod(EE)

度量指标：培养计划（CSE 或 EE）和生产率（LOC/Hour）

2. 原假设 H_0：是否拥有 C 语言相关的背景知识，对学生们编程过程中产生的每千行（每 1000 行代码）错误个数没有影响。

原假设 H_0：每千行代码中的错误率与是否拥有 C 语言经验无关。

备择假设 H_1：每千行代码中的错误率与 C 语言经验相关。

度量指标：C 语言经验和每千行代码中的错误率（Faults/KLOC）。

上述假设意味着需要收集以下数据：

- 培养计划：用 CSE 或 EE 度量（定类尺度）
- 生产率：通过代码行数/开发时间来度量。因此，需要度量程序的大小（代码行数需要根据编码标准和计数标准来衡量）和开发时间（开发程序需要的分钟数）。在计算生产率时，需要将开发时间转化为以小时为单位。值得注意的是，本实验中研究的是总的程序代码量（10 个编程作业的代码量的和）和开发 10 个程序所用的总开发时间。因此，也就意味着本实验并不研究单个编程作业。

  代码的行数是通过使用一个计数程序进行计量的（定比尺度）。被纳入统计的行是新的或是修改过的代码行。

  因此，开发时间采用分钟度量（定比尺度），生产率也是采用定比尺度进行度量的。
- C 语言经验是通过将学生先前的 C 语言经验分为四个级别（定序尺度）来度量
164 的。这四个级别分别是：

（1）没有先验经验。

（2）读过一本书或上过一门课程。

（3）一些工业开发经验（小于 6 个月）。

（4）工业开发经验（超过 6 个月）。

因此，C 语言经验是使用定序尺度来度量的。

- 错误率（Faults/KLOC）是通过错误数量除以代码行数来度量的。

最后，这些假设和度量给统计检验类型的选择带来了一些约束。虽然度量标度决定了能够应用何种统计方法，但我们也可能出于某些其他原因希望放宽这些限制条件。对于这个问题，我们将会在后面讨论实验的实际设计类型时进一步探讨。

12.2.3　变量选择

培养计划和 C 语言经验是独立变量。而生产率和每千行代码的错误率是非独立变量。

12.2.4　主体甄选

实验主体是基于便利原则进行选择的，即本实验的主体是参加 PSP 课程的学生。这些学生可以看作是这两种培养计划下所有学生的一个样本，但并非一个随机样本。

12.2.5　实验设计

在以上小节中，我们已经阐明了研究问题且已选定了独立变量与非独立变量；同时，也已确定了这些变量的度量标度。因此，现在可以着手设计实验了。设计实验的第一步是声明总体设计原则：

随机策略　实验对象不是随机分配给实验主体的。所有的学生都使用 PSP 方法和它的 10 个任务。研究目标也不是用 PSP 方法和其他方法进行比较。实验主体亦如上文所述选择了学习 PSP 课程的学生而非随机选择。此外，10 个任务的顺序也不是随机 165
的。作业的顺序并不重要，因为评估使用的度量值是 10 段开发程序的结果。

分块阻断　实验并没有使用系统的方法进行分块。将 10 段程序作为一个整体进行度量评估而不是对每段程序单独进行度量评估的决策可以看作是为了阻断这 10 段程序间的差异，也就是说，阻断程序间的差异对实验结果的影响。

均衡设计　本来最好是能够有一个平衡的数据集，但由于实验研究是基于一门参与者学习的课程，因此，不可能影响参与者的知识背景，从而无法平衡数据集。

标准设计类型　参照第 8 章介绍的各种设计类型标准衡量上述信息，可以发现，上述实验设计可以被归类到标准设计类型中。同时，本书还提供了可以使用的统计检验方法。

（1）第一次评估的定义、假设和度量意味着其设计类型为：“单因子双处置”。其中，因子是指培养计划，而处置则是指 CSE 和 EE。由于采用定比尺度度量非独立变

量，因此适合使用参数检验。在本案例中，选择使用 t-检验。

（2）第二种设计属于“单因子多处置”的设计类型。此时，因子指 C 语言经验，多种处置是指评价经验等级的四种类型。由于采用定比尺度度量非独立变量，因此也适合使用参数检验。在本案例中，选择使用 ANOVA 检验进行评估。

12.2.6　实验工具

受试个体的背景和经验通过第一次课程上的问卷调查可以获得，参见附录 A 中的表 A-1。表 A-1 中的数据代表着学生的特征，因此，将作为实验中的独立变量。研究的对象是在 PSP 课程中开发出的程序。在 PSP 课程过程中，文献［82］提供了指南及度量方法。

12.2.7　有效性评价

本实验需要考虑四种有效性威胁。*内部有效性*（Internal validity）主要聚焦于实际研究的有效性。*外部有效性*（External validity）可以从以下几个方面分析：Lund 大学中将来准备参加 PSP 课程学习的学生、Lund 大学的学生（或者更实际地指 CSE 和 EE 的学生）、一般的 PSP 开发人员和一般的软件开发人员。*结论有效性*（Conclusion validity）关
166 注实验处置与实验结果之间的关系以及得出结论的能力。*结构有效性*（Construction validity）关注将实验结论泛化为实验背后的理论。

课堂内的内部有效性可能不存在问题，大量的测试样本数（同学生人数相同）确保了良好的内部有效性。

对外部有效性而言，如果在 Lund 大学使用相似的方法授课，很可能会获得相似的结果。如果希望将这些结果泛化到其他学生，比如那些没有参加过课程学习的学生，将更困难一些。由于他们来源于不同人群，可能对软件开发没有兴趣。实验的分析结果可能也会推广到其他 PSP 课程，只要能够基于参与者的背景（计算机科学或电气工程）以及他们在某种特定程序语言的经验对参与者进行比较即可。

对结论有效性而言，最大的威胁是 PSP 课程期间搜集的数据的质量。由于期望学生在课程中将这些数据作为作业的一部分提供，因此，存在数据造假的风险或者由于失误而产生错误数据的风险。同时，由于研究问题本身与学生背景无关，因而可能存在的数据不一致性也与特定的学生背景无关。因此，结论有效性并不关键。

结构有效性面临两种主要威胁。第一种威胁是定义的度量方法可能不适合对研究对象进行度量。例如，“代码行数/开发时间”是一种度量生产率的好方法吗？第二种对结构有效性的威胁来源于实验是课程的一部分，将根据学生在课程中的表现给学生打分。这意味着学生可能会偏向于给出他们认为能够让他们获得更好分数的数据。但事实上，在课程开始我们就强调了学生成绩与实际的数据无关，而是取决于及时、正确地提交作业和课堂上所提交的报告中表达的对学习内容的理解。

虽然得到的结论来自于针对 PSP 课程展开的实验，但这些结果也可以推广到一般

的软件开发过程。没有理由证明，来自于不同培养计划或者拥有某种特定编程语言经验的人在执行 PSP 和一般的软件开发过程时会有不同的表现。但如果考虑参与者的背景差异（存在差异大小的区别），则“他们在执行 PSP 和一般的软件开发过程时会有不同的表现”很可能是合理的。重要的问题是，确实存在差异，但差异的大小是次要的。

12.3　操作

12.3.1　准备

实验主体（学生）并不清楚将针对哪方面开展实验研究。他们被告知研究人员希望研究不同背景的参与者学习 PSP 课程的结果，但并没有告诉他们实验假设。从学生的观点来看，他们主要是在选修一门课程而不是参与一项实验。我们向所有学生保证 167
会匿名使用信息。

调查问卷已经提前准备好，其他大多数材料来自于 PSP 教材［82］。

12.3.2　执行

整个实验历时 14 周，在此期间，学生们定期提交了 10 个编程作业。实验数据主要通过表格收集。在课程结束时，使用面谈法对课程和 PSP 方法进行评估。

如前所述，实验是在一个大学的 PSP 课程中进行的。因此，不允许实验影响 PSP 课程的教学目标。同单纯的 PSP 课堂教学存在的主要差异是，对学生背景的初始调研。

12.3.3　数据确认

共收集了 65 个学生的数据信息。课程结束后，课程涉及的所有成员共同分析了学生取得的成绩。其中，有 6 个学生的数据被认定为不合理或至少是有问题的而被移除。在此阶段，是否去除学生的数据并不是基于对其真实数据的评估，而是基于我们是否相信学生提交的数据并相信数据是否具有代表性。这 6 个学生的数据被移除的原因是：

- 两个学生的数据没有被正确地填入表格中。
- 一个学生完成作业的时间比其他学生晚得多，并且其在 PSP 课程上同样有很长一段时间没有完成工作。这可能会影响数据。
- 两个学生没有按时提交他们的作业并且比其他学生要求得到更多的帮助，因此，认定这些额外的帮助可能影响他们的数据。
- 最后，有一个学生的背景和其他人完全不同。因此，他的数据也被移除。

因此，从 65 个学生中移除了 6 个学生，即留下了 59 个学生的数据进行统计分析 168
和结果解释。

12.4　分析与解释

12.4.1　描述性统计

作为数据分析的第一步，使用描述性统计将收集的数据可视化。

学生来源 VS 生产率　图 12-1 展示了按照学生人数与生产率类别分类的分属两个培养计划的学生的生产率。第一类是生产率为每小时编程 5 到 10 行，依此类推，第八类是每小时编程 40 至 45 行。通过图 12-1 可以看出 EE 的学生生产率较低。此外，可以明显地看出，CSE 的学生的生产率差异更大。我们共有 32 个 CSE 的学生和 27 个 EE 的学生。CSE 学生的平均生产率是 23.0，标准差为 8.7；而 EE 的学生的平均生产率为 16.4，标准差为 6.3。为了能够更好地理解所获得的数据，我们绘制了箱形图，如图 12-2所示。

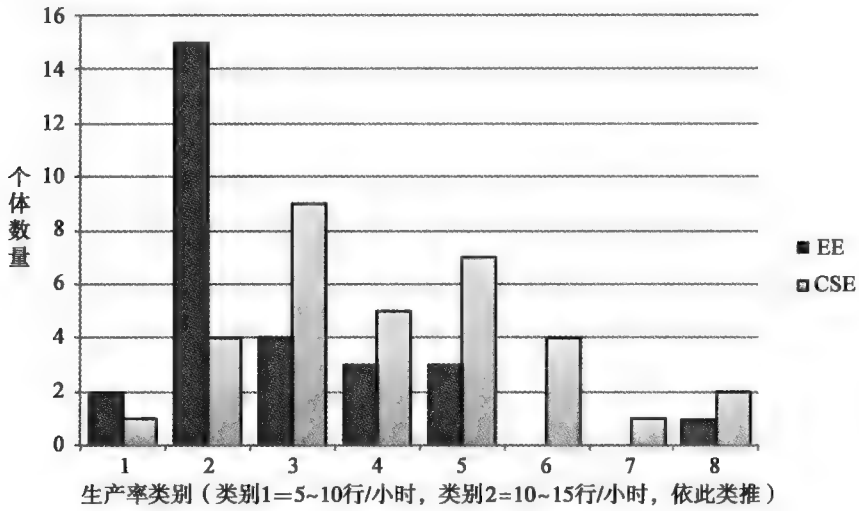


图 12-1　生产率频率分布（分类显示）

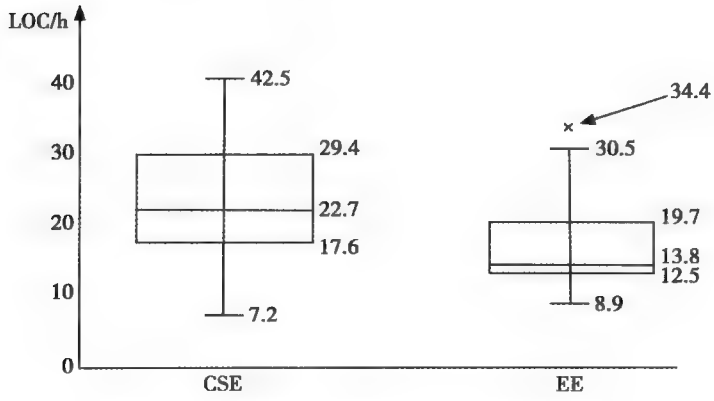


图 12-2　两种培养计划的学生生产率的箱形图

箱形图的延长线是根据 Frigge 等［60］中的方法绘制的，并已在第 10 章进行了讨论。延长线的长度是箱体长度的 1.5 倍，是分别在其上确界和下确界上加上或减去 1.5 倍的四分位距（即箱体长度）后得到的。例如，对于 CSE 的学生来说（如图 12-2 所示）：中位数 =22.7，箱体长度 =29.4 −17.6 =11.8，上延长线的长度为：29.4 +1.5 ×11.8 =47.1。但箱体图构造规则要求，上延长线顶端和下延长线尾端不应该超过数据集中的最大值和最小值。因此，上延长线的顶端取数据集的最大值 42.5。此规则是为了避免负值或其他类型的不合理值的出现。图 12-2 中的其他值可以使用与此类似的方式来确定。

从图 12-2 中不难看出，EE 的学生生产率较低。因此，也许可以通过假设检验识别出其中的统计差异。以下使用 t-检验进行检验。

与上延长线和下延长线相比，观察离群点同样很重要。对于 CSE 的学生，没有超出延长线的离群点。对于 EE 的学生，存在一个值 34.4 超出了延长线。由于我们认为它只是一个非正常值而不是一个极端值，故而还是决定将其纳入分析，而没有将其视为离群点。

C 语言经验 VS. 错误率（Faults/KLOC）　每类不同类别 C 语言经验的学生数如表 12-1 所示，同时呈现的还有均值、中位数以及各个类别的标准差。

表 12-1　不同类型 C 语言经验学生的错误率（Faults/KLOC）

学生类型①	学生数量	Faults/KLOC 的中位数	Faults/KLOC 的均值	Faults/KLOC 的标准差
1	32	66.8	82.9	64.2
2	19	69.7	68.0	22.9
3	6	63.6	67.6	20.6
4	2	63	63.0	17.3

169 ~ 170

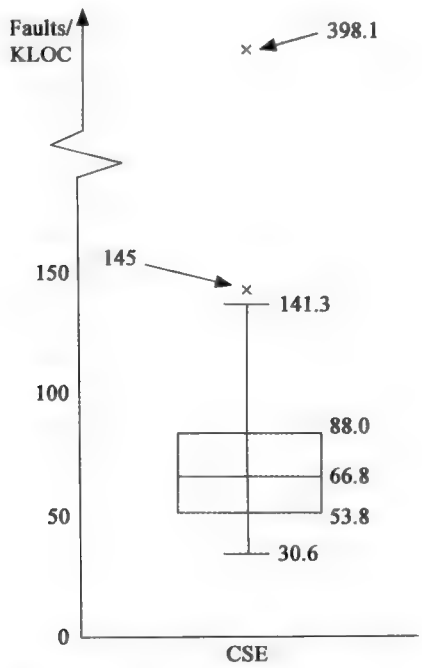
①不同经验的分类解释见 12.2.2 节。

从表 12-1 可以看出，分布向没有或很少有 C 语言经验的方向倾斜。如果参考 Faults/KLOC 的均值，似乎经验更多的学生犯的错误更少。在根据假设对均值和中位数进行比较时，中位数的变化与期望不同。标准差，特别是类别 1 的标准差非常大，建议进行更深层次的调查。因此，也可以用箱形图来分析此数据集。

我们对这四个类别的数据分别建立了箱形图。其中类型 2 ~4 的箱形图没有异常，其值均落在了延长线内，且上延长线顶端和下延长线底端分别与最大最小值相等。而类型 1 的箱形图却显得相当耐人寻味，如图 12-3 所示。

如图 12-3 所示，下延长线的底端与 Faults/KLOC 的最小值相等。但上延长线顶端却与 Faults/KLOC 的最大值不同，存在两个非正常值：145 和 398.1。后者是一个极端值，比最小值大十多倍，也几乎是第二大值 145 的三倍。因此，可认为此极端值导致了标准差增大。为检验第二个假设，我们使用了 ANOVA 检验。

使用描述性统计能更好地理解数据内涵，包括如何理解假设检验的结论以及如何发现离群点可能造成的潜在问题。



171

图 12-3　类别 1 的 Faults/KLOC 箱形图

12.4.2　数据约简

数据约简常常伴有争议性，因为只要有数据被丢弃，就会造成信息丢失。有两种不同的数据约简方法：

- 移除单个数据点，如离群点。
- 对数据进行分析，基于分析结果发现其中一些变量高度相关，从而将相关的度量组合构成某种更抽象的度量。

这意味着我们可以移除一些数据点或者减少变量的个数。在移除数据点时，被移除的主要是离群点。这并不意味着所有的离群点都应该被移除，但是它们确实是选择移除的候选对象。应该牢记的是，不要仅仅因为数据点与信念或假设不符而简单地将之移除。但从另一个角度看，去除那些使合理关系变成不合理的数据点至关重要，比如，去除一个在重复实验中无法再现的、极端的离群点。

为了约简变量个数，需要使用专门进行数据约简的统计学方法，比如主成分分析法和因子分析法［90，91，118］。由于此处的目标不是约简变量的个数，因此，将不在此讨论这些方法。

由于我们总是倾向于获得或证明某种结论，从而倾向于去除导致结论不能成立的数据点，因此最好能对数据集如何约简制定一些约束。因此，对上述数据，我们仅仅移除了 Faults/KLOC 这个指标上的极端离群点。移除此点后，类型 1 的数据如表 12-2 所示。

表 12-2　C 语言经验为类别 1 的学生的 Faults/KLOC

学生类型	学生数量	Faults/KLOC 的中位数	Faults/KLOC 的均值	Faults/KLOC 的标准差
1	31	66	72.7	29.0

当然，离群点的移除减小了均值和标准差。虽然类别 1 的 Faults/KLOC 的均值仍然是最大的，但同其他三个类别相比差异已经不那么大了。在从第二个数据集中约简一个数据点后，已不可能对其执行统计检验了。

12.4.3　假设检验

认为“CSE 的学生生产率更高”的第一个假设使用 t-检验进行检验，而运用 ANOVA 检验则被用来评估“C 语言经验越多意味着 Faults/KLOC 越小”这个假设。 172

培养计划 VS. 生产率　t-检验（非成对双侧检验）的结果如表 12-3 所示。

由表 12-3 可知，假设 H_0 被拒绝，即来源于不同培养计划的学生在编程效率方面具有显著差异。p 值（p-value）非常小，故而结果非常显著。造成差异的实际原因需进一步评估。

表 12-3　t-检验结果

因　子	平均差异	自由度	t-值	p-值
CSE vs. EE	6.1617	57	3.283	0.0018

表 12-4　ANOVA 检验结果

因子：C vs. Faults/KLOC	自由度	方差和	平均方差	F-值	p-值
处置间	3	3483	1160.9	0.442	0.7236
错误	55	144304	2623.7		

C 语言经验 VS. Faults/KLOC　该假设采用 ANOVA 来检验，表 12-4 展示了检验结果。

虽然从均值这方面可以观察到一些差异，但统计分析的结果并不显著，表明具有不同类别 C 语言经验的学生在其编写的程序中 Faults/KLOC 间并无显著差异。

由于级别 3、4 中包含的学生的数量非常有限，因此我们将 2～4 三个级别归为一组，以研究级别 1 与其他级别学生之间的差异。采用 t-检验评估是否能够将级别 1 与其他级别学生区分开来，结果不具备显著性。

12.5　总结

我们研究了两个假设：

（1）培养计划 vs. 生产率；

（2）C 语言经验 vs. Faults/KLOC。

结论表明，CSE 的学生生产率更高。虽然我们没有在假设中正式声明，但这一结论符合预期。我们的预期是基于这样的认识：大多数来自于 CSE 的学生比 EE 的学生选修过的计算机科学与软件工程方面的课程更多。

我们无法从统计学角度证明学生在 C 语言方面的经验会影响其每千行代码的错误
173 率。有趣的是，Humphrey［82］曾建议应该在 PSP 课程之前选修 C 语言这门广为人知的程序设计语言，从而使得学习者能够集中精力学习 PSP 而不是程序设计语言。实验获得的结果也许预示以下的一个或多个结果：

- 随着学生人数的增加，其中的差异将会更加显著。
- 错误的数量并不显著受参与者的前期经验影响。也许在开发软件时人们会犯一定数量的错误。错误的类型可能不同，但错误总量趋于相同。
- 没有此语言经验的学生也可能编写过大量程序，这可能直接影响了 Faults/KLOC。

当然，我们也可以找到其他解释，但它们都有同一个共同的特点即需要重复。重复对于使我们理解、继而控制和改进软件开发方法至关重要。此外，其他因素同样也需要研究。

从有效性方面看，有理由相信 CSE 的学生（从总体来说）比来自于其他学科的学生生产率更高。这或多或少源于其教育背景，不值得惊讶。

既然不能给出程序设计语言的经验同每千行代码错误率之间存在任何统计意义上的关系，也就没有什么结论可以推而广之。这表明我们还需要通过重复 PSP 实验或者在其他环境中进行类似实验来进行进一步研究。

12.6　结论

由于上述实验比较的因素不是随机分配给实验主体的，而是实验主体本身固有的属性（如实验主体的教育背景），因此，它是一个准实验。将学生作为实验主体，为结论提供了好的内部有效性，但却是以影响结论外部有效性为代价的。由于实验是在 PSP 课程中进行的且其情境都已明确定义，因此，对于进行重复实验非常有利。

实验历时了若干周。这可能成为结构有效性面临的最大威胁。然而，鉴于此实验是一个准实验，此威胁的力度有所降低。学生没有机会在教育背景上造假。虽然已经告诉学生会在将来使用他们收集的数据，但没有告诉他们具体使用方式，因此，可以不受他们偏好的影响。

在分析阶段，由于有 6 个学生没有跟随实验进程提交数据，这 6 个样本数据点被移除。另有 3 个数据点由于位于箱形图延长线外而被作为离群点进行了分析，其中只
174 有 1 个因为严重影响标准差而导致实验结果偏差被移除。

第13章
Experimentation in Software Engineering

视角间真有差异吗？基于场景的需求文档阅读的进一步实验[⊖]

背景 本章介绍了一篇在国际期刊上发表过的实验研究论文。此外，它还非常适合作为实践审阅技术的研究论文。应该注意到，这篇论文被期刊《经验软件工程》发表之前已经通过审稿并根据审稿意见进行了修订。这就意味着，虽然一般论文在第一次发表后也会通过不断修改而提高水平，但是该论文的质量高于一般的实验论文水平。如何评审科学论文会在附录 A.2 中进行详细说明。

摘要 基于视角的阅读（Perspective-Based Reading，PBR）是一种基于场景的审查技术，多个评审员从不同的视角（例如，用户、设计人员、测试人员）阅读同一篇文档。阅读是根据为每个视角特设的一个场景进行的。PBR 背后的基本假设是，不同的视角能发现不同的缺陷，如果阅读量相同，组合多个视角发现的缺陷比单一视角发现的缺陷更多。这篇论文分析了各种视角的差异，是先前研究的部分重复。实验在一个学术环境中进行，使用研究生作为实验主体。每个视角都应用了特定的建模技术：用户视角采用了案例建模技术，测试人员视角采用了等价类划分方法，设计人员视角采用了结构化分析方法。在实验中，30 个实验主体被划分为 3 组，即每个视角有 10 个实验主体。分析结果表明：三个视角在缺陷检出率和每小时发现的缺陷数方面并无显著差异；三个视角的缺陷覆盖率没有显著差异；仿真研究表明，采用所选择的统计检验方法，30 个主体就足够发现相对较小的视角差异。上述分析结果表明，组合多种视角的阅读方法相比于单一视角的阅读方法，可能并不会有更高的缺陷覆盖率。但是，在未来仍然有必要对此进行进一步研究以加深对视角差异的理解。

13.1 引言

需求文档的确认常常是人工进行的，这是因为需求文档中往往包含着需要一个期望软件系统做什么的非规范的表达。审查是一种由 Fagan 提出的、常见的用于软件文档的人工确认技术［54］。审查技术可以采用不同的方式实施，并且可以在整个软件开发过程中使用，以（1）增进理解、（2）发现缺陷，并（3）作为决策制定的依据。审查可以用于在开发过程早期发现缺陷。已有研究表明，该技术具有显著的成本效益（参见 Doolan［45］）。

⊖ 本章最初发表在 Empirical Software Engineering: An International Journal, Vol. 5, No. 4, pp. 331 – 356 (2000)。

审查过程的核心部分是缺陷检测（defect dection），由个体审查者阅读文档并记录缺陷（是准备的一部分，参见 Humphrey［81］）。自由检查（Ad Hoc）、检查单和基于场景的阅读（Scenario-based reading）是三种常用的缺陷检测技术。自由检查是一种没有提供指导的非结构化技术，即评审者根据他们的个人知识和经验检测缺陷。检查单则提供了一个议题和问题的清单，包含着以前审查的知识，试图帮助评审者在阅读时聚焦。而在基于场景的阅读方法中，不同的评审者具有不同的职责，并且通过特定的场景指导他们的阅读；同时，在阅读时要构建模型，而不仅仅是被动地阅读。

这里的“场景”（scenario）⊖指的是评审者需要遵从的脚本或程序。研究人员已提出了两种基于场景的阅读方法的变体：基于缺陷的阅读（Defect-Based Reading，DBR）［137］和基于视角的阅读（Perspective-Based Reading，PBR）［18］。前者（后文中用 DBR 指代）关注特定的缺陷类别，后者（后文中用 PBR 指代）关注文档使用者的视角。

审查过程的另一个任务是将缺陷汇编成一个统一的缺陷列表。该列表汇编了所有评审者发现的缺陷。这一步骤可能会包括移除误报的缺陷（将不是缺陷的内容报告为缺陷）和发现新的缺陷。该步骤通常在由一组评审人员参加的审查会议中进行。这种群组会议的有效性已在 Votta［175］、Johnson 和 Tjahjono［87］的文章中进
176 行了研究。

这篇论文描述了使用 PBR 方法进行基于场景的阅读的研究。该文采用经验型研究方法，论述了一个学术环境下正式的析因实验。该实验是此领域中曾经做过的一个实验的部分重复，其关注点在于针对 PBR 中视角间差异的精化假设。该论文关注于个体评审者的缺陷检测，而不包含有关群组会议的部分。

论文的结构如下。13.2 节通过概述以前做过的使用基于场景阅读的方法进行需求审查的实验结论，给出了一个相关工作的综述；13.3 节通过问题陈述给出了本文的研究动机；13.4 节介绍了实验计划、讨论了实验的有效性威胁；13.5 节报告了实验的操作；分析的结果在 13.6 节中给出；13.7 节给出了对结果的解释；13.8 节是总结和结论。

13.2　相关工作

现有的关于经验软件工程的文献包含了大量与审查相关的研究。正式实验是其中一种相关的研究策略［178］。本文中论述的实验与先前关于使用基于场景方法的审查实验相关。对于这些基于场景的审查实验的发现总结如下：

（1）Maryland-95 研究［137］在学术环境下比较了 DBR、自由检查和检查单的审查效果。该实验执行了两次，每次有 24 个实验主体参与。实验所用到的需求文档是一

⊖ 这里有术语混淆的风险，术语场景（scenario）也用于在需求工程中表示待开发系统的设想使用情景中的一系列事件。一个用例（use case）通常覆盖一组相关的（系统使用）场景。但在基于场景的阅读中，术语场景是一个元级概念，表示审查过程中一个文档的阅读者应该遵从的程序。

个水位监测系统（WLMS，24 页）和一个汽车巡航控制系统（CRUISE，31 页）的文档。

结果 1：使用 DBR 方法的评审员的缺陷检出率显著高于使用自由检查方法和检查单方法的评审员。

结果 2：使用 DBR 方法的评审员在特定类型缺陷的检出率显著高于其他两种方法；而在其他类型缺陷的检出率方面，与其他两种方法相同。这些特定类型的缺陷是指场景设计针对的缺陷。

结果 3：使用检查单方法的评审员的缺陷检出率没有显著高于使用自由检查方法的评审员。

结果 4：缺陷收集会议对缺陷检出率没有纯粹的提升作用——会议产生的积极效果被其产生的消极效果抵消。

（2）NASA 研究［18］在工业环境下对 PBR 与自由检查进行了比较。该实验由两部分组成：第一部分有 12 个实验主体参加，第二部分有 13 个实验主体参加。实验中用到了两组需求文档：一般需求文档和 NASA 需求文档。一般需求文档由自动取款机文档（ATM，17 页）和停车场控制系统文档（PG，16 页）组成；NASA 需求文档是两个飞行动力学需求文档（每个 27 页）。 177

结果 1：对于一般文档，使用 PBR 方法的个体的缺陷检出率显著高于使用自由检查方法。

结果 2：对于 NASA 类文档，使用 PBR 方法的个体的缺陷检出率没有显著高于使用自由检查方法。

结果 3：对于一般文档，使用 PBR 方法的模拟团队的缺陷检出率显著高于使用自由检查方法。

结果 4：对于 NASA 类文档，使用 PBR 方法的模拟团队的缺陷检出率显著高于使用自由检查方法。

结果 5：经验多的评审员没有更高的缺陷检出率。

（3）Kaiserslautern 研究［34］在学术环境下比较了 PBR 方法与自由检查方法，使用了 NASA 研究［18］中的 ATM 和 PG 文档。该实验执行了两次，分别有 25 个实验主体和 26 个实验主体参与。

结果 1：对于一般文档，使用 PBR 方法的个体的缺陷检出率显著高于使用自由检查方法。

结果 2：对于一般文档，使用 PBR 方法的模拟团队的缺陷检出率显著高于使用自由检查方法。

结果 3：不同视角下五种缺陷类别的缺陷检出率没有显著差异。

（4）Bari 研究［61］在学术环境下比较了 DBR、自由检查和检查单，使用了 Maryland-95 研究中的 WLMS 和 CRUISE 文档。该实验执行了一次，有 30 个实验主体参与。

结果 1：相较于自由检查和检查单方法，DBR 方法没有显著较高的缺陷检出率。

结果2：使用DBR方法的评审员对特定缺陷的缺陷检出率没有显著高于使用其他两种方法的评审员；在其他缺陷的缺陷检出率方面，使用DBR方法与其他两种方法也相同。特定缺陷是指场景设计专门针对的缺陷。

结果3：使用检查单方法的评审员的缺陷检出率没有显著高于使用自由检查方法的评审员。

结果4：缺陷收集会议对缺陷检出率没有纯粹的提升作用——会议产生的积极效果被产生的消极效果所抵消。

（5）Trondheim研究［164］比较了NASA研究中的PBR与一个改进的PBR（以下记为PBR2）。在PBR2中，为评审员提供了更多的关于如何进行基于视角阅读的指导。该研究在学术环境下进行，使用NASA研究中的ATM和PG文档。实验执行了一次，有48个实验主体参与。

结果1：使用PBR2方法的评审员的缺陷检出率没有显著高于使用PBR方法。

178

结果2：使用PBR2方法的个体评审所花费的时间显著长于使用PBR方法的个体。

结果3：使用PBR2方法的个体提出的潜在缺陷数目显著少于使用PBR方法的个体。

结果4：使用PBR2方法的个体的生产率和效率显著低于使用PBR方法的个体。

（6）Strathclyde研究［124］在学术环境下比较了DBR和检查单，使用了Maryland研究中的WLMS和CRUISE文档。该实验执行了一次，有50个实验主体参与。

结果1：对于WLMS文档，DBR方法的缺陷检出率没有显著高于检查单方法。

结果2：对于CRUISE文档，DBR方法的缺陷检出率显著高于检查单方法。

结果3：缺陷收集会议对缺陷检出率没有纯粹的提升作用——会议产生的积极效果被产生的消极效果抵消。

（7）Linköping研究［147］在学术环境下比较了DBR方法和检查单方法，使用了Maryland研究中的WLMS和CRUISE文档。总缺陷清单中加入了更多的缺陷。实验执行了一次，有24个实验主体参与。

结果1：对于WLMS文档，使用DBR方法的评审员的缺陷检出率没有显著高于使用检查单方法的评审员。

结果2：对于CRUISE文档，使用DBR方法的评审员的缺陷检出率没有显著高于使用检查单方法的评审员。

（8）Maryland-98研究［152］在学术环境下比较了PBR与自由检查，使用了Maryland研究中的ATM和PG文档。该实验执行了一次，有66个实验主体参与。

结果1：使用PBR方法的评审员的缺陷检出率显著高于使用自由检查方法的评审员。

结果2：经验丰富的个体使用PBR方法的缺陷检出率没有显著[⊖]高于使用自由检查方法的评审员。

⊖ Maryland-98研究的结果2－4的显著性水平是0.10，在其他结果中的显著性水平是0.05。

结果 3：经验中等的个体使用 PBR 方法的缺陷检出率显著高于使用自由检查方法的评审员。

结果 4：经验较少的个体使用 PBR 方法的缺陷检出率显著高于使用自由检查方法的评审员。

结果 5：使用 PBR 方法的个体的生产率显著低于使用自由检查方法的个体。

（9）Lucent 研究［138］在工业环境下重复了 Maryland-95 研究。该研究有 18 个 Lucent 科技的专业开发人员参与。此次重复实验是成功的，完全证实了 Maryland-95 研究的结果。 179

这些研究的结果有很大不同。Hayes［74］对 Maryland-95、Bari、Strathclyde、Linköping 和 Lucent 研究的结果进行了元分析，试图对这些实验和重复实验中获得的知识系统地进行讨论。元分析表明，审查方法的效果在不同实验中是不同的。Maryland-95 和 Lucent 研究的结果最为相似，元分析指出了将这两个研究区别于另外三个研究的特征：（1）这两个研究中的主体对使用的表示法更加熟悉；（2）这两个研究是在美国进行的，而其他三个研究是在欧洲进行的，同时，汽车中的巡航控制在美国比在欧洲应用得更普遍。现有的数据不可能检验这些假设，因此，还需要进行更多实验。

表 13-1 对以上研究进行了总结。Maryland-95、NASA、Kaiserslautern、Maryland-98 和 Lucent 研究表明基于场景的方法有更高的缺陷检出率。然而，Bari、Strathclyde 和 Linköping 的研究却不能印证这些结果，这就需要进一步的研究以增加对基于场景阅读的理解。

表 13-1　研究总结

研　究	目　的	环　境	主　体	是否显著
Maryland-95	DBR vs. 自由检查和检查单	学术	24 +24	是
Bari	DBR vs. 自由检查和检查单	学术	30	否
Strathclyde	DBR vs. 检查单	学术	50	不确定
Linköping	DBR vs. 检查单	学术	24	否
Lucent	DBR vs. 自由检查和检查单	工业	18	是
NASA	PBR vs. 自由检查	工业	12 +13	是
Kaiserslautern	PBR vs. 自由检查	学术	25 +26	是
Trondheim	PBR vs. PBR2	学术	48	否
Maryland-98	PBR vs. 自由检查	学术	66	是

以上很多研究表明，真实的团队会议在缺陷检测方面是不起作用的。（除了缺陷检测之外，召开群组会议当然还有很多其他好的理由，如建立共识、共享能力和制定决策等。）

本文介绍的研究在后续介绍中用 Lund 研究来命名。Lund 研究是 NASA 研究的一个部分重复实验。它基于 Maryland 大学提供的一个实验包，以支持基于场景阅读的经 180

验型研究。Lund 研究的研究动机将在下一节中给出。

13.3　研究问题

13.2 节对之前的研究进行了总结，这些研究主要关注于从缺陷检出率的角度比较基于场景阅读、检查单与自由检查技术；而 Lund 研究的目标是研究基于场景的阅读技术背后的基本假设“从不同视角能够发现不同的缺陷”是否成立。本研究的另一个关注点在于不同视角的效率，即每小时发现缺陷数。因此，研究问题可以归结如下：

（1）不同的视角能否检出不同的缺陷？

（2）一种视角是否会比另一种视角更优越？

优越性体现在两方面：*效果*（effectiveness），即现有缺陷中有多少能被发现（检出率）；*效率*（effeciency），即单位时间内能检出多少缺陷。

Basili 等人提出的视角有设计人员、测试人员和用户视角。用户是软件开发过程中重要的涉众，在需求抽取、分析和文档化阶段尤为重要。PBR 中的用户角色关注检测与系统使用相关的高抽象层次的缺陷，设计人员则关注内部结构中存在的缺陷，测试人员则关注验证阶段中存在的缺陷。

之前的研究主要关注从缺陷检出率的角度评判效果。对软件工程师而言，评估效率（如单位时间检出的缺陷数）也很重要，因为这个因素对于从业者决定是否引进一种新的阅读技术至关重要。具体的项目约束、应用领域约束和对所需工作量的估计都将成为权衡质量和成本的依据。

PBR 的一个主要目标是通过不同视角检出不同类型的缺陷，从而使得评审者之间的重复工作最小化。因此，自然就产生了一个问题：不同视角的评审者是否真的能够发现不同的缺陷。如果他们发现的缺陷相同，则无法最小化重复工作，PBR 的意图也就没有达到。如果所有的视角发现的是同样类型的缺陷，可能的原因有：（1）基于场景的阅读方法是不适合的；（2）与它们关联的场景无法充分支持这些视角；或者（3）还需要从其他视角分析以获得较大的覆盖差异。理想的解决方案是使用无重叠的视角和缺陷检出率尽可能高的视角进行审查，以使得 PBR 方法高度可靠且有效。Lund 研究通过调查不同视角能否检出不同缺陷来判断这些视角间是否存在重叠。

从缺陷内容估计的角度看，研究问题 1 也是值得关注的。用于进行缺陷内容估计的捕获－再捕获方法（capture-recapture approach）使用评审者发现的缺陷中重叠部分的多少来估算一个软件产品中剩余缺陷的数量［51，120］。为了研究捕获－再捕获估
181 算法应用在 PBR 审查中的效果，Thelin 和 Runeson［167］研究了在承认 PBR 的前提假设下，在 PBR 中使用捕获－再捕获方法的鲁棒性。在 Lund 研究中，研究了 PBR 的前提假设是否成立的问题。因此，Lund 研究和 Thelin 与 Runeson 的研究［167］互为补充地回答了捕获－再捕获估算法是否可以用于 PBR 审查的问题。

13.4　实验计划

本节描述了阅读实验的计划。计划包括独立变量和非独立变量的定义、实验中要检验的假设、实验设计、实验工具以及对实验有效性威胁的分析［178］。

阅读实验在一个与工业环境相近的学术环境中进行。实验主体是 Lund 大学 CSE 和 EE 就读硕士学位的四年级学生。

13.4.1　变量

独立变量决定了如何对非独立变量取样。实验的目的是通过在两个实验对象（需求文档）上应用不同的阅读视角和审查方法，研究其中的差异。审查对象与 Maryland 大学实验包中的对象相同，实验设计和工具也基于这个实验包。研究中的变量如表 13-2 所示，表中有简短的解释。

表 13-2　变量

	变 量 名	变 量 值	描　述
独立变量	PERSP	{U, T, D}	每个主体使用其中的一个视角：用户（U），设计人员（D），测试人员（T）
	DOC	{ATM, PG}	审查对象是两个需求文档：自动取款机（ATM）和停车场控制系统（PG）文档。ATM 文档有 17 页包含 29 个缺陷。PG 文档有 16 页包含 30 个缺陷
控制变量	EXPERIENCE	定序型	不同视角的实验主体经验值采用一个五级定序尺度度量，在分配主体视角时使用（详见 13.4.3 节和 13.6.4 节）
非独立变量	TIME	整型	每个评审员在个人准备阶段花费的时间，由实验主体记录。时间单位为分
	DEF	整型	每个评审员发现的缺陷数量，不包括伪真值。为了保证对所有的候选缺陷是平等的，实验人员去除了伪真型缺陷
	EFF	60 × DEF/TIME	缺陷发现速率，即每小时发现的缺陷数，按（DEF × 60）/TIME 计算
	RATE	DEF/TOT	缺陷发现效果，即发现的缺陷数量与总缺陷数量的比值（也叫缺陷检出率），由 DEF 除以已知的缺陷总数得到
	FOUND	整数	在特定文档中发现了某个特定缺陷的属于某个视角的评审员的数量。该变量用于分析不同视角的缺陷发现分布

13.4.2　假设

基于视角的阅读方法有效的前提假设是，由于不同的评审员使用不同的视角阅读，使得发现的缺陷重复更少从而能更有效地审查［18］。本研究的目标在于使用经验型方法检验该假设是否成立。因此，与不同视角的性能相关的假设陈述如下。三个原假

设分别讨论了视角的效率、效果和分布。

- $H_{0,EFF}$：假设不同视角有相同的缺陷发现效率，即不同视角每小时发现的缺陷数没有差异。
- $H_{0,RATE}$：假设不同视角有相同的效果或缺陷检出率，即不同视角发现的缺陷与总缺陷之比没有差异。
- $H_{0,FOUND}$：假设不同视角发现的缺陷一样，即不同视角发现的缺陷分布是相同的。

182
~
183

13.4.3　实验设计

为了检验这些假设，我们使用一个有两个因素（PERSP 和 DOC）的析因设计实验。实验设计如表 13-3 所示。实验针对两个文档和三个视角展开。

表 13-3　实验设计

		视　角		
		用　户	设计人员	测试人员
文档	ATM	5	5	5
	PG	5	5	5

与 NASA 研究［18］相似，实验主体的视角分配（U，D，T）是根据实验主体报告的经验（见 13.6.4 节）来划分的。这种基于经验分配视角的方式是为了确保每个视角的评审员经验分布均衡，从而使实验的输出反映不同视角的影响而非主体经验的影响。针对经验的调查问卷需要实验主体针对他们在每个视角上的经验打分，分为 5 个等级。然后将主体进行 3 次排序，对每个视角给出一个有序表，经验等级高的主体排名靠前；经验等级相同的主体在组内被随机排序。然后，按照以下规则将主体分配到不同的视角：从 3 个视角的排序序列中随机选择一个视角的排序序列作为开始，从该视角的序列顶端选取一个主体，赋予该视角，并将此主体从其余两个列表中去除；按照轮转法，轮流选择下一个视角，并依此循环，直到所有的主体都被分配到某种视角之下。

阅读实验的工具包括两个需求文档以及时间和缺陷的报告模板。实验工具源于 Maryland 大学的实验包，并在重用时包含尽可能少的修改。

上文描述的析因设计使用描述统计学（柱状图和箱形图）进行分析，对假设 $H_{0,EFF}$ 和 $H_{0,RATE}$ 进行方差分析（ANOVA）［125］；对于假设 $H_{0,FOUND}$ 使用卡方检验［157］和相关性分析［144］。

13.4.4　有效性威胁

实验结果的有效性取决于实验设置中的因素。根据实验目标，不同类型的有效性优先级不同。在本案例中，我们分析了 4 类有效性［37，178］：结论有效性、内部有

效性、结构有效性和外部有效性。 184

结论有效性（Conclusion validity）　关注结果的统计分析和主体的构成。本实验中采用的是众所周知的统计技术，它们对于假设的违背具有鲁棒性。结论有效性的一个普遍威胁是样本数量少。样本数量少可能会削弱从数据中揭示模式的能力，特别是卡方检验的样本很少，我们将在 13.6.3 节对此进行详细说明。

内部有效性（Internal validity）　关注不包括研究人员的知识在内的可能影响涉及因果关系的独立变量的事项。本实验存在两种内部有效性威胁：选择和工具。本实验是软件工程课程的一个必修环节，因此主体甄选不是随机的，这会对实验的有效性产生威胁。所使用的需求文档也可能会影响实验结果。文档有相当的缺陷倾向并且文档中的其他问题也可能会被当作缺陷。另一方面，为了便于比较，本实验与对比实验最好能够使用相同的缺陷定义。其他因素对内部有效性的威胁影响都很小。由于实验主体只对一个对象采用一种方案进行审查，因此，不存在成熟风险（即实验主体针对相同实验对象采用多种方案或利用同一方案对多个实验对象审查时，由于学习效应而导致结果失真的风险）。在审查期间，主体使用不同的视角，但是视角间的差异不足以大到怀疑不同方案间会存在补偿平衡或补偿竞争。同时，告知实验主体，他们的课程成绩与他们在实验中的表现无关，只取决于他们是否认真地参与了实验。但这样一来，就存在着实验主体缺乏积极性的问题。例如，他们可能会认为参加实验是浪费时间或者学习技术的积极性不高。然而，主导实验课程的教师已经采取措施尽力激励学生，课程明确规定认真参与实验是通过课程考核的必备条件。课程教师认为学生在审查中都很认真。

结构有效性（Construct validity）　关注将实验结果泛化为实验背后的概念或理论的能力。对结构有效性的一个主要威胁是选择的视角或视角的阅读技术对于基于场景的阅读来说可能不具有代表性或者不好。这限制了对这些特定的视角和技术做出结论的适用范围。其他因素对结构有效性的威胁影响很小。实验主体不知道所设定的假设，并且没有参与任何关于 PBR 优点和缺点的讨论，因此他们无法猜测出应该期望得到什么结果。

外部有效性（External validity）　关注于将实验结果类推到此次实验以外的环境中的普遍适应性。对外部有效性最大的威胁是使用学生作为实验主体。然而，使用即将完成学业开始工作的四年级学生可以减轻这一威胁。虽然实验设置旨在使实验与真实的审查情景类似，但是实验主体参与的过程不是真实软件开发项目的一部分。审查对象也力图与实际项目一致，但所使用的文档仍然相当短，真正的软件需求文档可能 185
会包含更多页。然而，由于审查过程和文档与真实案例非常相似，我们认为实验设置和审查对象对外部有效性的威胁是有限的。

如上所述，实验中存在对结构有效性、内部有效性和外部有效性的威胁，但这些威胁在先前的研究也同样存在。因此只要实验得出的结论没有超出这些威胁的限制范围，结果就是有效的。

13.5　实验操作

本实验是在1998年春天进行的。每个学生参与了一个2小时的入门课程，讲述了研究概览和缺陷分类。学生们都完成了关于经验的调查问卷，并按照13.4.3节所述，为每个学生即实验主体分配了一个特定的视角。同时通知学生，实验是课程的必修部分，但分数只与参与实验的认真程度有关而与学生的个人表现无关，并确保学生的匿名性。

随后，进行了2小时的练习。在练习中介绍了三个PBR视角并且用一个视频租赁系统（VRS）的需求文档进行了例示。在练习的第二个小时里，实验主体从各自的视角出发利用VRS需求文档练习了该视角的阅读技术，其间如有疑问可以提出。练习时也对数据收集表格进行了解释，并在练习时使用。课后，学生们自己完成了VRS文档的基于视角的阅读。

练习时分发了实验用的讲义，包括以下指导工具。

（1）缺陷分类：用缺陷列表的方式描述缺陷类别。

（2）时间记录日志：记录阅读花费的时间。

（3）缺陷表：记录发现的缺陷。

（4）阅读指南：分别针对用户、设计人员和测试人员视角设计的阅读指南。

（5）建模表格：分别针对用户、设计人员和测试人员视角设计的建模表格。

（6）需求文档：ATM或PG的需求文档。

要求学生不讨论ATM或PG文档，也不讨论他们发现的缺陷。在真正开始数据收
186 集前，允许他们基于VRS文档讨论PBR视角。

13.6　数据分析

本节展示了针对所收集数据的统计分析。数据来源于实验主体提交的表格。每个主体的缺陷日志中的每个缺陷都与Maryland大学实验包中提供的原始“正确”的缺陷表进行了比较。通过会议，作者们对每个缺陷进行了探讨以判定它是否对应于一个“正确”的缺陷。如果没有找到相应的“正确”缺陷，则认为报告的缺陷是伪真的[⊖]。同时收集了报告中的时间花费，并且计算了EFF、RATE和FOUND测度。总数据集如表13-6～表13-8所示。

13.6.1　不同视角的个体表现

个体表现的箱形图[⊜]如图13-1所示，以视角PERP和文档DOC为划分，依据每小

⊖ 如果Maryland实验室包中的缺陷列表不完整，一些被标为伪真的缺陷可能是真正的缺陷。但从重复实验的角度看，使用相同的“正确”缺陷列表很重要，因此，决定使用原始的“正确”缺陷列表。而由于只有极少数伪真的缺陷值得质疑，因此，认为这个决定对结果没有任何显著影响。

⊜ 箱形图中矩形的高度位置分别对应数据的上四分位数和下四分位数，矩形中部的线段代表中位数。上下延长线的端点对应着第10和第90百分位的数值大小。

时发现的缺陷数（EFF）和发现缺陷与总缺陷之比（RATE）两个指标绘制。

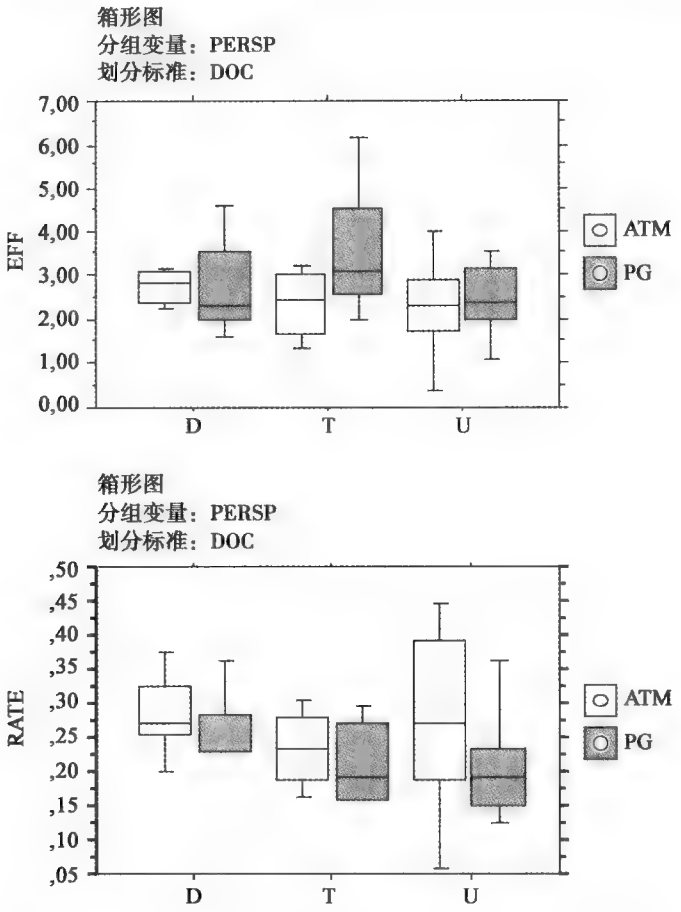


图 13-1　按 DOC 和 PERSP 划分的各组 EFF 和 RATE 的箱形图

对指标 EFF 而言，从测试人员视角审查 PG 文档得到的 EFF 均值高于用户和设计人员视角；但对于 ATM 文档而言，设计人员视角获得的 EFF 均值更高。对指标 RATE 而言，在审查这两个文档时，设计人员视角都比另外两个视角有更高的 RATE 均值。但由于每组中的数据点太少，无法根据离群点和偏度对箱形图做更进一步的解释。

当度量几个非独立变量时，可以使用多变量方差分析（MANOVA）来评估均值集合中是否存在统计上的显著差异。针对视角效果的 MANOVA 检验能够表明不同视角效果间没有显著差异，但却缺少对交互影响的分析。此外，如表 13-4 和表 13-5 所示，通过 ANOVA 方差分析，PERSP 变量的 EFF、RATE 均值均无显著差异。这些分析表明，对于三个视角中的任何一个视角而言，EFF 和 RATE 的原假设都不能被拒绝。

表 13-4　EFF 的 ANOVA 方差分析表

	DF	Sum of Sq	Mean Sq	F-Value	p-value	Lambda	Power
PERSP	2	1.751	0.875	0.737	0.4893	1.473	0.156

（续）

	DF	Sum of Sq	Mean Sq	F-Value	p-value	Lambda	Power
DOC	1	1.640	1.640	1.380	0.2516	1.380	0.193
PERSP * DOC	2	2.229	1.114	0.937	0.4055	1.875	0.187
Residual（残差）	24	28.527	1.189				

表 13-5　RATE 的 ANOVA 方差分析表

	DF	Sum of Sq	Mean Sq	F-Value	p-value	Lambda	Power
PERSP	2	0.012	0.006	0.802	0.4602	1.604	0.166
DOC	1	0.011	0.011	1.488	0.2344	1.488	0.205
PERSP * DOC	2	0.004	0.002	0.259	0.7739	0.518	0.085
Residual（残差）	24	0.172	0.007				

187
~
188

13.6.2　不同视角发现的缺陷

本节着重研究关于不同视角发现的缺陷间重合度的假设 $H_{0,FOUND}$。柱状图 13-2 展示了其描述性统计量。对于每个文档，每个视角发现的缺陷数量的分布都显示在图中。

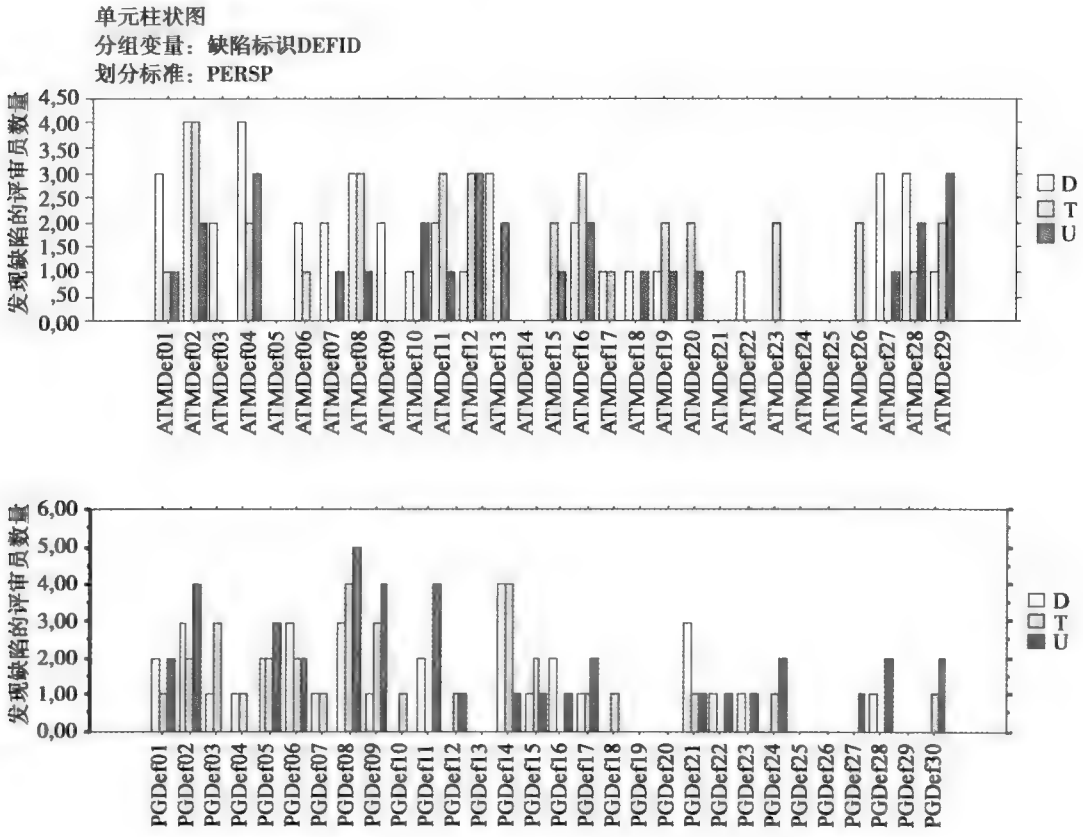


图 13-2　展示发现每个缺陷的评审者数量分布的柱状图

不同视角的分布看起来没有特定的模式；每个视角发现的缺陷都相似地散布于缺陷空间中。如果不同视角的分布间存在大的差异，则柱状图中会出现一些缺陷组：对于某个视角而言，能够较多地发现这类缺陷；而对于其他视角而言，就只能发现较少的这类缺陷。

为了比较每个视角所发现缺陷的分布和探究不同视角发现的缺陷是否存在显著差异，我们用卡方检验创建了一个列联表，如图 13-3 所示。三个视角都没有发现的缺陷被排除在列联表之外（见图 13-3 中的“包含条件”），因为这些缺陷对检验差异性没有帮助。

DEFID,PERSP汇总表
包含条件：PG数据中Counts>0

Num.Missing	0
DF	46
Chi Square	33,951
Chi Square P-Value	,9058
G-Squard	.
G-Squared P-Value	.
Contingency Coef.	,494
Cramer's V	,402

DEFID,PERSP汇总表
包含条件：ATM数据中Counts>0

Num.Missing	0
DF	46
Chi Square	41,676
Chi Square P-Value	,6538
G-Squared	.
G-Squared P-Value	.
Contingency Coef.	,535
Cramer's V	,448

DEFID,PERSP的观测频率
包含条件：PG数据中Counts>0

	D	T	U	Totals
PGDef01	2	1	2	5
PGDef02	3	2	4	9
PGDef03	1	3	0	4
PGDef04	1	1	0	2
PGDef05	2	2	3	7
PGDef06	3	2	2	7
PGDef07	1	1	0	2
PGDef08	3	4	5	12
PGDef09	1	3	4	8
PGDef10	0	1	0	1
PGDef11	2	0	4	6
PGDef12	0	1	1	2
PGDef14	4	4	1	9
PGDef15	1	2	1	4
PGDef16	2	0	1	3
PGDef17	1	1	2	4
PGDef18	0	1	0	1
PGDef21	3	1	1	5
PGDef22	1	0	1	2
PGDef23	1	0	1	2
PGDef24	0	1	2	3
PGDef27	0	0	1	1
PGDef28	1	0	2	3
PGDef30	0	1	2	3
Totals	33	32	40	105

DEFID,PERS的观测频率
包含条件：ATM数据中Counts>0

	D	T	U	Totals
ATMDef01	3	1	1	5
ATMDef02	4	4	2	10
ATMDef03	2	0	0	2
ATMDef04	4	2	3	9
ATMDef06	2	1	0	3
ATMDef07	2	0	1	3
ATMDef08	3	3	1	7
ATMDef09	2	0	0	2
ATMDef10	1	0	2	3
ATMDef11	2	3	1	6
ATMDef12	1	3	3	7
ATMDef13	3	0	2	5
ATMDef15	0	2	1	3
ATMDef16	2	3	2	7
ATMDef17	1	1	0	2
ATMDef18	1	0	1	2
ATMDef19	1	2	1	4
ATMDef20	0	2	1	3
ATMDef22	1	0	0	1
ATMDef23	0	2	0	2
ATMDef26	0	2	0	2
ATMDef27	3	0	1	4
ATMDef28	3	1	2	6
ATMDef29	1	2	3	6
Totals	42	34	28	104

图 13-3　每个文档中被 U、T、D 发现的缺陷的卡方检验和列联表

卡方分布的 p 值很不显著，这表明用这种检验方法和这些数据集无法展现不同视角发现的缺陷间的差异。文［157，pp. 199－200］给出了一些使用卡方分布检验必须满足的前提条件，认为预期频次少于 5 的单元格不多于 20% 且不存在预期频次少于 1 的单元格时可以使用卡方分布。本案例中的数据集不满足这些前提条件，但是这些前提条件也可能太苛刻。因为本案例中的预期频率是均匀分布的，卡方检验可能依然有效（详见 13.6.3 节）。

189
~
190

卡方检验无法度量差异的程度。为了分析视角间的差异度（或相似度），需要使用相关性分析（如图 13-4 所示），使用皮尔逊（Pearson）相关系数进行分析［143，pp. 338－340］。

ATM文档
相关性分析

	相关性	p值	低于95%	高于95%
用户，测试人员	,480	,0076	,138	,720
用户，设计人员	,499	,0052	,162	,732
测试人员，设计人员	,258	,1789	-,120	,570

此次计算中有29个观测值

相关性分析
包含条件：ATM-ctable.data中User>0或Tester>0或Designer>0

	相关性	p值	低于95%	高于95%
用户，测试人员	,357	,0867	-,054	,665
用户，设计人员	,352	,0915	-,059	,662
测试人员，设计人员	,043	,8449	-,367	,439

此次计算中有24个观测值

PG文档
相关性分析

	相关性	p值	低于95%	高于95%
用户，测试人员	,463	,0092	,123	,706
用户，设计人员	,543	,0016	,228	,756
测试人员，设计人员	,601	,0003	,307	,790

此次计算中有30个观测值

相关性分析
包含条件：PG-ctable.data中User>0或Tester>0或Designer>0

	相关性	p值	低于95%	高于95%
用户，测试人员	,319	,1300	-,097	,640
用户，设计人员	,414	,0438	,012	,700
测试人员，设计人员	,493	,0134	,112	,748

此次计算中有24个观测值

图 13-4　每个文档中视角的相关性分析

对每个文档进行了两种不同的相关性分析：一种分析针对了所有的“正确”缺陷，另一种分析只针对那些被至少一个评审者发现过的缺陷。我们认为按照后者分析更加合适，因为我们关注的是分析不同视角发现的缺陷集间的差异。没有被发现的缺

陷对研究视角间的差异没有贡献。

p 值表示相关系数是否显著，置信区间表示相关系数可能的区间。

相关性分析表明各视角间显著正相关，这也意味着当一个视角发现一个缺陷时，其他视角很可能也发现了该缺陷。只有审阅 ATM 文档的设计人员和测试人员之间的相关性不显著。 191

另一种定性分析视角间的重叠性的方法是维恩图（Venn-diagrams），NASA 研究中使用了该方法［18，p. 151］。

为了比较，我们用 Lund 研究的数据做了维恩图，如图 13-5 所示。每个被发现过的缺陷都根据其被哪些视角发现过而分到 7 类中相应的一类中。维恩图中的数值表示每类中有多少个缺陷。例如，对于 PG 文档，有 10 个缺陷是三个视角都发现的，有 5 个缺陷被用户视角和设计人员视角发现，只有 1 个缺陷是仅被用户视角发现的。

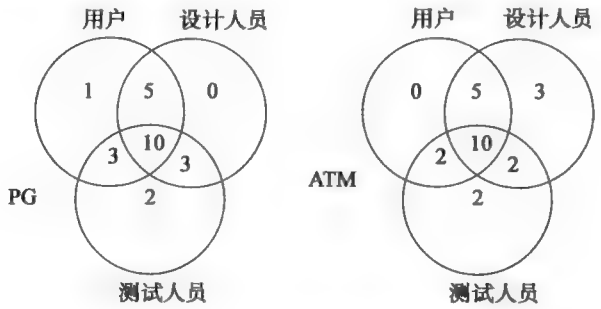


图 13-5　PG 和 ATM 文档中缺陷覆盖的情况

这种分析方式对主体数量非常敏感。很可能，某一个评审员发现某一个缺陷就足以改变该缺陷所属的分类。一个缺陷被发现的可能性随着评审员数量的增加而增长，并且如果评审员数量很多，很可能所有的缺陷都被包含在被所有视角发现的那一类之中。这意味着这种分析方式的鲁棒性不强，并且无法提供对一般案例的有意义的解释。在本例中，我们至少可以说，图 13-5 中的缺陷覆盖分析与之前的结果不矛盾，即我们不能拒绝原假设：不同视角发现的缺陷集是相似的。如图 13-5 所示，被 3 个视角都发现的缺陷是最大的一类。

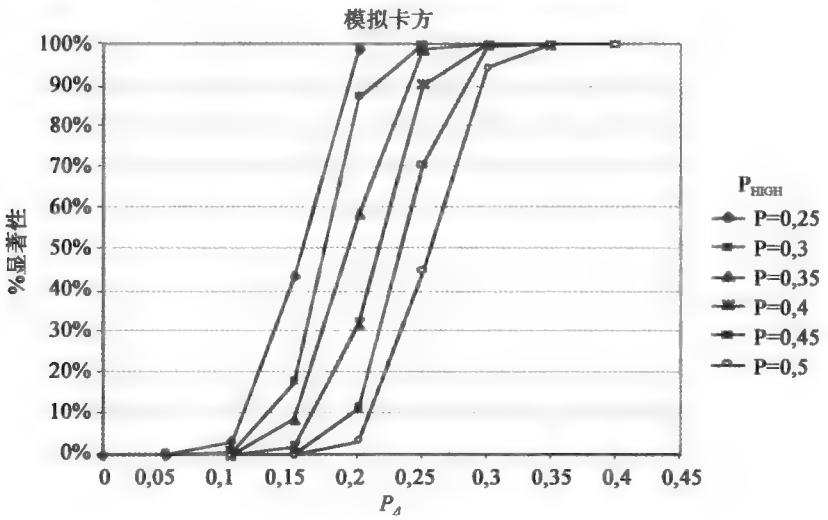
13.6.3　样本空间足够大吗

Lund 研究结果表明视角间没有显著差异，那么到底是由于数据缺乏差异还是统计检验无法发现差异呢？比如，由于数据的数量有限而无法发现差异。为了评估卡方检验结论是否合理，我们采用随机变化法来模拟不同视角的缺陷检测数据，并用卡方检验对其进行检验。 192

模拟模型是依照之前章节介绍的实验设计来设计的。不同仅在于模拟实验中某个视角检出某种缺陷的可能性是独立变量；同时，由于没有建模时间属性，非独立变量只有 FOUND。具体的模型设计如下：

- 每个模拟文档中的缺陷数是30。
- 对每次模拟审查，每个视角都使用10个评审员。假定一个文档包含三种不同类型的缺陷，不同类型的缺陷被检出的概率不同。某种视角能高概率地（P_{HIGH}）检出其中三分之一的缺陷，而低概率地（P_{LOW}）检出另外三分之二的缺陷。P_{HIGH}与P_{LOW}间的差值记为P_{Δ}。概率差值按步长为0.05取区间0.05到0.5之间的值。概率差值的值域是根据Lund研究中的度量均值确定的。
- 每个审查模拟1000次。

使用卡方检验来检验假设$H_{0,FOUND}$，结果如图13-6所示。每次模拟实验单独检验。该图展示了每个模拟案例被拒绝的检验次数所占的比例。对于所有P_{Δ}大于0.3的模拟案例，该检验可以显著地展示出模拟视角间的差异。对于所有P_{HIGH}小于0.25的模拟案例，如果P_{Δ}大于0.2，则差异性就展现出来。检验的显著性水平是0.05。模拟研究表明，即使视角间的差异很小且样本空间很小，卡方检验也可以发现FOUND上的差异。



193

图13-6　关于$H_{0,FOUND}$显著性检验结果的片段

13.6.4　主体经验

通过问卷调查来度量主体的经验。该问卷调查了主体在一般意义下在每个视角上的经验以及从三个视角使用特定建模技术（案例建模、等价类划分与结构化分析）的经验。对这两种经验均采用5级定序尺度度量：1=没有经验，2=在课程或书中学过，3=在课堂项目中练习过，4=在工业项目中使用过，5=在多个工业项目中使用过。

图13-7展示了每个主体在其所分配的视角方面所具备的平均经验，既包括一般经验，也包括具备特定建模技术的经验。

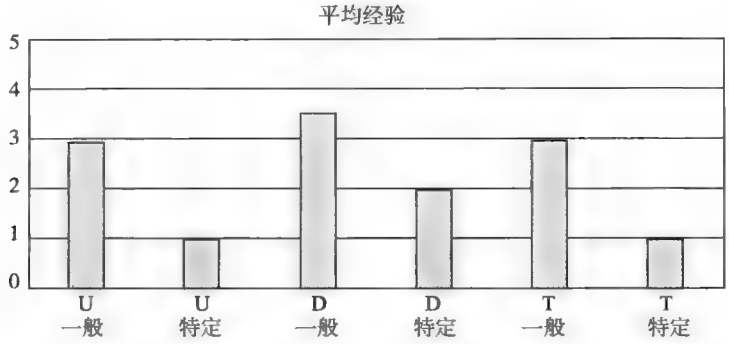


图 13-7　主体的平均经验，考虑到对视角的一般经验和对特定建模技术的经验

可以看出，如预期的一样，主体的分配（依据 13.4.3 节中的算法描述）可以使各视角的经验相对均衡。还必须注意到，学生的产业实践经验很少。

13.7　结果解释

在本节中，我们将根据 13.4.2 节中的假设对数据进行分析解读。前两个假设采用 ANOVA 检验，第三个假设采用卡方检验。以下三个原假设都不能被拒绝：

- $H_{0,EFF}$假设不同视角每小时发现的缺陷数相同。不能拒绝此假设。
- $H_{0,RATE}$假设不同视角发现的缺陷数相同。不能拒绝此假设。
- $H_{0,FOUND}$假设不同视角发现的缺陷相同。不能拒绝此假设。

194

因此，我们可以得出结论：三个视角（用户、设计人员与测试人员）之间没有显著差异。这对于以上三个假设而言都是正确的，即缺陷发现的效果或效率间均无显著差异。此外，不同视角审查缺陷时所花费的时间也没有显著差异，即使用何种技术并不影响缺陷审查所需的时间。如果上述结果可以复制和泛化，那么不同视角间的差异缺乏将严重影响 PBR 方法存在的基础。由于 PBR 方法的优势就在于假定不同视角关注不同类型的缺陷，因此能够检出不同的缺陷集。但本研究表明，三个视角发现的缺陷集之间并无统计意义上的显著差异，因此 PBR 方法的优势将会受到质疑。

对结果的结论有效性威胁是样本数量少，尤其对于卡方检验而言更是如此。然而模拟实验表明：对 30 个根据各视角进行缺陷审查的主体而言，即使主体间缺陷发现概率差异相对较小，卡方检验也可以发现该差异。此外，不同视角所发现的缺陷的柱状图（如图 13-2 所示）也没有显示任何显著的模式，这也从另一个角度支持了上述无显著差异的结果。在可接受的范围内的 ANOVA 统计表明，不同视角之间也没有任何显著差异。当我们试图将结果泛化到一般的基于场景的阅读中时，特定视角和针对该视角的阅读技术可能会对结果的有效性造成威胁。

关于主体积极性的有效性威胁可以通过比较 Lund 研究和其余研究的缺陷检出率来评估。NASA 研究［18］中预备实验和主实验中个体的 PBR 检出率均值分别为 0.249 和 0.321，Lund 研究中个体的 PBR 检出率均值为 0.252。这些比率具有可比性，能够

支持 NASA 研究中的主体和本研究中的主体同样被激励的假设。

根据本结论的特点，没有考虑 13.4.4 节中提到的其他对于有效性产生威胁的因素。

13.8　总结和结论

本研究重点关注采用基于视角阅读的方法（PBR）审查需求文档，利用 Maryland 大学［19］的实验包在学术环境中进行，是对之前实验的部分复制。

研究目标包含如下两个方面：

（1）在效果（缺陷检出率）和效率（每小时发现缺陷数）方面探究不同视角的性
195 能差异。

（2）探究不同视角间缺陷覆盖率的差异，并评估 PBR 的基本假设：不同视角能发现不同缺陷。

实验设置包括两个需求文档和为三个视角设置的场景（用户使用案例建模，设计人员使用结构化分析，测试人员使用等价类划分）。总共有 30 个理科硕士生被分成 3 组、每 10 个主体一个视角参与了实验。

数据分析结果总结如下：

（1）在缺陷检出率、每小时发现的缺陷数方面，用户、测试人员、设计人员三个视角间没有显著差异。

（2）三个视角间所检出的缺陷覆盖率没有显著差异。

对这些结果的解释表明：与单视角的阅读审查相比，多视角组合的阅读可能并不具备更高的缺陷覆盖率。

这些结果与 PBR 的主要假设矛盾。13.2 节中总结的一些已有研究结果表明，基于场景的阅读和自由检查相比有显著优势，但目前还没有研究对不同视角间的性能差异进行统计分析。此外，13.2 节中的已有研究也没有关注效率（每小时发现的缺陷数），而只是专注于将缺陷检出率作为主要的非独立变量来研究。从软件工程的视角来看，一个方法的花费和效率都应该是核心关注点，因此，不仅要研究 PBR 的缺陷检出率，还要研究其在条件有限时是否能够很好地执行。

还有一些对结果有效性的威胁，包括：

（1）实验设置可能不现实。

（2）视角可能不是最佳的。

（3）实验主体可能积极性不高或训练不足。

（4）主体的数量可能过少。

基于以下理由，我们认为对有效性的威胁是可控的：（1）审查对象与工业需求文档相似；（2）这些视角是从软件工程过程的角度选取的；（3）实验主体是大学四年级学生，这些学生对软件工程感兴趣并且基于自身兴趣参与了这门选修课程；另外，许多公司有很大一部分员工是新入职的（即大学四年级学生与公司新进的员工本质差异

不大）；（4）模拟研究表明，用选定的分析方法分析给定的等量数据，能够发现视角间相当微小的差异。

像这样的一个单一实验，还不足以改变人们对 PBR 的看法。对已有的实验数据进行同样的分析和以评估视角间的差异为目的进行重复实验，可以使 PBR 技术的优点和缺点更清晰明了，也可以更好地控制有效性威胁。 196

13.9　个体表现数据

表 13-6　每个主体的数据

Id	视　角	文　档	时　间	缺　陷	效　率	检 出 率
1	U	ATM	187	8	2.567	0.276
2	D	PG	150	8	3.200	0.267
3	T	ATM	165	9	3.273	0.310
4	U	PG	185	11	3.568	0.367
5	D	ATM	155	8	3.097	0.276
6	T	PG	121	8	3.967	0.267
7	U	ATM	190	7	2.211	0.241
8	D	PG	260	7	1.615	0.233
9	T	ATM	123	6	2.927	0.207
10	U	PG	155	6	2.323	0.200
11	D	ATM	210	11	3.143	0.379
12	T	PG	88	9	6.136	0.300
13	U	ATM	280	11	2.357	0.379
14	D	PG	145	11	4.552	0.367
15	T	ATM	170	5	1.765	0.172
16	U	PG	120	6	3.000	0.200
17	D	ATM	190	9	2.842	0.310
18	T	PG	97	5	3.093	0.167
19	U	ATM	295	2	0.407	0.069
20	D	PG	180	7	2.333	0.233
21	T	ATM	306	7	1.373	0.241
22	U	PG	223	4	1.076	0.133
23	D	ATM	157	6	2.293	0.207
24	T	PG	130	6	2.769	0.200
25	U	ATM	195	13	4.000	0.448
26	D	PG	200	7	2.100	0.233
27	T	ATM	195	8	2.462	0.276
28	U	PG	125	5	2.400	0.167
29	D	ATM	200	8	2.400	0.276
30	T	PG	150	5	2.000	0.167

197

13.10　各视角发现缺陷的数据

13.10.1　文档 PG

表 13-7　个体阅读文档 PG 所发现的缺陷标识 D#(1) 或没有发现的缺陷标识 D#(0)

	用户视角						测试人员视角						设计人员视角					
D#	2	8	14	20	26	S	4	10	16	22	28	S	6	12	18	24	30	S
1	0	0	1	0	1	2	1	0	0	0	0	1	1	0	0	1	0	2
2	1	1	1	0	1	4	1	0	1	0	0	2	0	1	1	1	0	3
3	0	0	0	0	0	0	1	1	0	1	0	3	0	0	0	1	0	1
4	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	1
5	0	0	1	1	1	3	1	0	0	0	1	2	0	1	1	0	0	2
6	1	1	0	0	0	2	0	1	1	0	0	2	1	0	0	1	1	3
7	0	0	0	0	0	0	1	0	0	0	0	1	0	1	0	0	0	1
8	1	1	1	1	1	5	1	1	1	1	0	4	1	1	0	1	0	3
9	1	1	1	1	0	4	1	1	1	0	0	3	0	1	0	0	0	1
10	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0
11	1	1	0	1	1	4	0	0	0	0	0	0	0	1	1	0	0	2
12	1	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	1	1	1	1	0	1	1	4	1	1	0	1	1	4
15	0	0	1	0	0	1	0	1	0	0	1	2	1	0	0	0	0	1
16	1	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0	2
17	0	0	0	1	1	2	1	0	0	0	0	1	0	1	0	0	0	1
18	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	1	0	0	1	0	0	0	0	1	1	1	0	1	0	1	3
22	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1
23	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1
24	0	0	1	1	0	2	0	0	0	0	1	1	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
28	0	1	1	0	0	2	0	0	0	0	0	0	1	0	0	0	0	1
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	1	0	1	0	0	2	0	0	1	0	0	1	0	0	0	0	0	0
S	8	7	11	7	7	40	11	6	6	4	5	32	8	9	5	6	5	33

13.10.2　文档 ATM

表 13-8　个体阅读文档 ATM 所发现的缺陷标识 D#(1) 或没有发现的缺陷标识 D#(0)

	用户视角						测试人员视角						设计人员视角					
D#	1	7	13	19	25	S	3	9	15	21	27	S	5	11	17	23	29	S
1	0	0	0	1	0	1	0	0	0	1	0	1	1	1	0	0	1	3
2	1	0	1	0	0	2	1	0	1	1	1	4	1	1	1	0	1	4
3	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	2
4	0	1	1	1	0	3	1	1	0	0	0	2	1	1	1	0	1	4
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0	2
7	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1	2
8	0	0	1	0	0	1	0	1	0	1	1	3	1	1	0	0	1	3
9	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	2
10	0	1	1	0	0	2	0	0	0	0	0	0	0	1	0	0	0	1
11	1	0	0	0	0	1	0	1	0	1	1	3	1	0	1	0	0	2
12	1	1	1	0	0	3	0	0	1	1	1	3	0	1	0	0	0	1
13	1	0	1	0	0	2	0	0	0	0	0	0	0	1	1	1	0	3
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	1	0	0	0	1	1	0	0	0	1	2	0	0	0	0	0	0
16	0	1	1	0	0	2	1	1	1	0	0	3	0	1	1	0	0	2
17	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1
18	0	0	1	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1
19	1	0	0	0	0	1	1	1	0	0	0	2	0	0	0	1	0	1
20	0	0	1	0	0	1	0	0	1	1	0	2	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
23	0	0	0	0	0	0	0	0	1	0	1	2	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	1	0	1	0	2	0	0	0	0	0	0
27	1	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	1	3
28	1	0	1	0	0	2	1	0	0	0	0	1	1	0	1	0	1	3
29	1	1	1	0	0	3	1	0	0	0	1	2	0	0	0	1	0	1
S	8	7	11	2	0	28	8	6	5	7	8	34	8	11	9	6	8	42

致谢　首先，感谢参与实验的所有学生。我们要特别感谢 Maryland 大学的 Forrest Shull，感谢其对 UMD 实验包提供的支持，并对本文的初稿给出了很多好的建议；感谢

匿名评审提供的所有建设性意见；感谢 Lund 大学通信系统系的 Claes Wohlin，Martin Höst 和 Håkan Petersson，感谢他们仔细审阅了本文；特别感谢 Lund 大学数学科学中心的 Anders Holtsberg，感谢其在统计分析上给予的专业帮助。本工作的部分资助由瑞典
200 国家工业和技术发展委员会（National Board of Industrial and Technical Development，NUTEK）提供，项目编号 1K1P-97-09690。

附　录

Experimentation in Software Engineering

附录 A
Experimentation in Software Engineering

练　　习

不同类型的练习说明可以在前言中找到。总之，本书的目标在于提供四种类型的练习。

理解型练习：这种练习旨在突出每一章中最重要的问题。如第 1-11 章节中的练习。

训练型练习：这种练习的目的在于鼓励实践实验，包括设置假设和进行统计分析。

回顾型练习：第 12 章与第 13 章提供了实验示例。这部分的目的在于帮助读者回顾与阅读已发布的实验。

任务型练习：这些练习是为了加深对如何在软件工程中使用实验来评价方法和技术的理解而设置的。

理解型的习题在每章结尾处都能找到，而其他三种类型的练习可以在此附录中找到。

A.1　训练

可以使用统计学程序包或者统计学书中的表完成练习。也可以使用附录 B 中的表格进行训练，但是表格中的数据只能在显著性水平为 5% 时使用，如果使用了其他的显著性水平，那么必须使用其他资源。应该注意的是，附录 B 主要是为第 10 章中的例子提供解释所用。

A.1.1　正态分布数据

第 10 章所用到的统计方法中最复杂的是正态分布的拟合优度检验，参见 10.2.12 节。为了确保能够正确使用该方法，请完成如下练习：

1. 参考表 10-20，在相同的数据集上将数据分为 12 段进行拟合优度检验。

A.1.2　实践

在第 12 章，根据选修 PSP 课程学生的背景对 PSP 课程的结果进行了比较。在第 12 章中介绍的只是部分分析，完整的数据集将会在表 A-2 和表 A-3 中提供。表 A-1 展示了第一节课发放的调查问卷，表 A-2 展示了调查结果。在表 A-3 中，通过以下 7 个方面对课程产出进行了度量：

- 规模（Size）：10 个程序中新增或修改的代码行数。
- 时间（Time）：10 个程序的总的开发时间。
- 生产率（Prod.）：每小时开发的代码行数。

- 错误（Faults）：10 个程序中标记的出错数，即所有找到的错误数，包括编译错误。
- 错误/每千行（Faults/KLOC）：每千行代码出现的错误数。
- 预期规模（Pred. Size）：预期程序规模的相对误差的绝对值。例如，对程序规模的预估无论是高估 20% 还是低估 20%，都用 20% 表示，没有任何指示符表明估计误差的方向。
- 预期时间（Pred. Time）：预期开发时间的相对误差的绝对值。

表 A-1 学生特征

方 面	描 述	答 案
学习计划（用 Line 表示）	答案：CSE 或 EE	
对计算机科学和软件工程方面的一般知识（用 SE 表示）	1. 很少，但对新课程很好奇 2. 不是我的专长（专注于其他科目） 3. 至少在其中某一方面相当不错，但不是我的主要精力所在 4. 我的研究重点	
程序设计方面的一般知识（用 Prog. 表示）	1. 只参加过 1-2 门课程学习 2. 参加过 3 门或更多门课程学习，但没有任何行业经验 3. 参加过少数课程学习，具备一些行业经验 4. 参加过超过三门课程学习，并有 1 年以上的行业经验	
PSP 方面的知识（用 PSP 表示）	1. 这是什么？ 2. 我曾经听说过 3. 大致了解 4. 我读过一些相关材料	
C 语言知识（用 C 表示）	1. 没有任何先验知识 2. 读过一本书或参加过课程学习 3. 有一些行业经验（少于 6 个月） 4. 有行业经验	
C++语言知识（用C++表示）	1. 没有任何先验知识 2. 读过一本书或参加过课程学习 3. 有一些行业经验（少于 6 个月） 4. 有行业经验	
课程数量（用 Courses 表示）	提供一组课程列表，要求学生根据他们是否学过该课程而回答 yes 或 no。此外，要求他们根据他们曾经阅读过的、与某特定课程相关的读物，补充课程清单	

根据第 12 章中的表述和表 A-2 和表 A-3 中的数据，回答以下问题：

1. 如何改进调查问卷？想一想，如何很好地度量背景、经验和能力。
2. 根据已有数据，参考第 12 章中的已有假设定义一些新的假设，并阐明定义这

些假设的动机。

3. 使用的是什么类型的抽样技术?

4. 分析你定义的假设，结果如何?

5. 讨论你的发现的外部有效性。这些结果能够泛化到 PSP 课程之外么? 这些结果能够泛化到工业界的软件工程师么?

表 A-2　背景问卷调查的相关信息

Subject	Line	SE	Prog.	PSP	C	C++	Courses
1	1	2	1	2	1	1	2
2	1	3	2	1	2	1	4
3	2	3	2	2	2	2	7
4	1	3	2	3	2	1	3
5	1	3	2	3	2	1	5
6	2	4	3	2	1	1	7
7	2	3	2	2	1	2	7
8	1	3	2	2	1	1	4
9	2	4	3	2	1	1	9
10	2	4	2	1	1	1	7
11	1	2	2	1	2	1	3
12	2	4	3	2	1	1	9
13	2	4	3	2	3	3	8
14	2	3	2	2	1	1	6
15	1	3	2	2	1	1	5
16	2	4	2	1	1	1	10
17	1	3	3	1	1	1	5
18	2	4	3	2	1	3	6
19	2	4	3	3	3	3	8
20	1	1	1	1	1	1	2
21	2	3	3	2	2	2	10
22	2	3	2	3	1	1	5
23	1	3	2	2	1	1	4
24	1	2	1	1	1	1	3
25	2	4	3	1	2	2	7
26	1	3	2	2	1	1	5
27	2	4	3	2	3	2	7
28	1	3	2	3	1	1	2
29	2	4	2	3	1	1	7
30	2	3	3	1	2	3	6

（续）

Subject	Line	SE	Prog.	PSP	C	C + +	Courses
31	1	3	2	2	2	2	5
32	2	3	3	1	2	2	10
33	2	4	3	1	1	1	5
34	1	2	2	1	2	2	3
35	1	2	1	1	1	1	2
36	1	2	1	2	1	1	2
37	1	2	2	2	2	2	2
38	2	4	2	2	2	1	6
39	1	2	1	2	1	1	2
40	2	4	3	1	4	4	7
41	2	3	3	2	2	2	8
41	2	4	3	2	2	2	9
43	1	3	2	1	1	1	3
44	1	4	3	2	3	2	7
45	2	4	2	2	2	1	6
46	2	2	4	2	4	4	7
47	2	4	3	2	3	2	7
48	1	2	2	2	1	1	2
49	1	3	3	1	1	1	3
50	2	3	2	3	1	1	8
51	2	4	2	4	2	2	8
52	2	4	3	3	3	2	8
53	2	4	3	3	2	2	10
54	1	2	1	2	1	1	2
55	1	2	2	2	1	1	4
56	2	3	2	1	1	1	8
57	1	2	3	1	1	1	4
58	2	4	3	3	1	1	6
59	1	2	2	2	2	1	4

表 A-3 PSP 课程的结果

Subject	Size	Time	Prod.	Faults	Faults/KLOC	Pred. size	Pred. time
1	839	3657	13.8	53	63.2	39.7	20.2
2	1249	3799	19.7	56	44.8	44.1	21.2
3	968	1680	34.6	71	73.3	29.1	25.1
4	996	4357	13.7	35	35.1	24.3	18.0

（续）

Subject	Size	Time	Prod.	Faults	Faults/KLOC	Pred. size	Pred. time
5	794	2011	23.7	32	40.3	26.0	13.2
6	849	2505	20.3	26	30.6	61.1	48.2
7	1455	4017	21.7	118	81.1	36.5	34.7
8	1177	2673	26.4	61	51.8	34.6	32.5
9	747	1552	28.9	41	54.9	51.0	18.2
10	1107	2479	26.8	59	53.3	22.6	14.0
11	729	3449	12.7	27	37.0	26.9	52.0
12	999	3105	19.3	63	63.1	26.0	19.8
13	881	2224	23.8	44	49.9	47.9	39.9
14	730	2395	18.3	94	128.8	63.0	20.3
15	1145	3632	18.9	70	61.1	33.3	34.8
16	1803	3193	33.9	98	54.4	52.9	21.8
17	800	2702	17.8	60	75.0	34.3	26.7
18	1042	2089	29.9	64	61.4	49.3	41.5
19	918	3648	15.1	43	46.8	49.7	71.5
20	1115	6807	9.8	26	23.3	34.1	22.4
21	890	4096	13.0	108	121.3	19.3	34.8
22	1038	3609	17.3	98	94.4	21.4	52.0
23	1251	6925	10.8	498	398.1	21.8	34.1
24	623	4216	8.9	53	85.1	40.5	36.3
25	1319	1864	42.5	92	69.7	43.7	45.0
26	800	4088	11.7	74	92.5	42.6	36.2
27	1267	2553	29.8	88	69.5	53.0	30.1
28	945	1648	34.4	42	44.4	33.3	17.9
29	724	4144	10.5	49	67.7	32.8	17.8
30	1131	2869	23.7	102	90.2	29.2	15.5
31	1021	2235	27.4	49	48.0	18.0	25.0
32	840	3215	15.7	69	82.1	85.6	54.0
33	985	5643	10.5	133	135.0	27.3	31.0
34	590	2678	13.2	33	55.9	83.0	20.0
35	727	4321	10.1	48	66.0	17.0	22.7
36	955	3836	14.9	76	79.6	33.3	36.8
37	803	4470	10.8	56	69.7	18.2	27.7
38	684	1592	25.8	28	40.9	35.0	34.1
39	913	4188	13.1	45	49.3	25.3	27.5
40	1200	1827	39.4	61	50.8	31.6	20.9

（续）

Subject	Size	Time	Prod.	Faults	Faults/KLOC	Pred. size	Pred. time
41	894	2777	19.3	64	71.6	21.3	22.4
42	1545	3281	28.3	136	88.0	35.0	16.1
43	995	2806	21.3	71	71.4	15.6	38.3
44	807	2464	19.7	65	80.5	43.3	26.4
45	1078	2462	26.3	55	51.0	49.1	51.6
46	944	3154	18.0	71	75.2	59.0	39.2
47	868	1564	33.3	50	57.6	50.4	45.2
48	701	3188	13.2	31	44.2	21.2	49.7
49	1107	4823	13.8	86	77.7	19.3	28.4
50	1535	2938	31.3	71	46.3	29.6	20.7
51	858	7163	7.2	97	113.1	58.4	32.9
52	832	2033	24.6	84	101.0	48.4	25.6
53	975	3160	18.5	115	117.9	29.5	31.5
54	715	3337	12.9	40	55.9	41.7	26.6
55	947	4583	12.4	99	104.5	41.0	22.3
56	926	2924	19.0	77	83.2	32.5	34.7
57	711	3053	14.0	78	109.7	22.8	14.3
58	1283	7063	10.9	186	145.0	46.5	26.6
59	1261	3092	24.5	54	42.8	27.4	45.3

A.1.3 程序设计

在一次实验中，20 个程序员编写同一个程序，其中 10 人使用语言 A，另外 10 人使用语言 B。与 B 语言相比，A 语言是新采用的语言。公司计划通过实验来判定语言 A、B 的优劣，如果 A 语言优于 B 语言，则以后公司将转而使用 A 语言。在开发的过程中，对程序大小、开发时间、所有移除的缺陷总数以及测试中移除的缺陷数等进行记录、度量。

为 20 个程序员随机分配一种编程语言，评估语言是否会对 4 种度量变量有影响。收集的数据（这些数据都是虚构的）如表 A-4 所示。

1. 本实验使用的是哪种实验设计？

2. 为评估定义假设。

3. 使用箱形图根据四种因素的居中趋势和离散程度调查语言之间的差异。是否存在离群点，如果有，是否应该被移除？

4. 假设可以使用参数检验，评估编程语言对四个度量变量的影响。可以从分析结果中得出什么结论？

5. 使用非参数检验评估编程语言对四个度量变量的影响。可以从分析结果中得出

什么结论？同使用参数检验的结果进行比较。

6. 讨论结果的有效性，并讨论使用参数检验是否恰当。

7. 如果参与的程序员是自己选择的编程语言，那么会对结果的有效性产生什么影响？结论仍然成立吗？

表 A-4　编程练习的数据

编程语言	程序大小（LOC）	开发时间（分）	总的缺陷数	测试缺陷数
A	1408	3949	89	23
A	1529	2061	69	16
A	946	3869	170	41
A	1141	5562	271	55
A	696	5028	103	39
A	775	2296	75	29
A	1205	2980	79	11
A	1159	2991	194	28
A	862	2701	67	27
A	1206	2592	77	15
B	1316	3986	68	20
B	1787	4477	54	10
B	1105	3789	130	23
B	1583	4371	48	13
B	1381	3325	133	29
B	944	5234	80	25
B	1492	4901	64	21
B	1217	3897	89	29
B	936	3825	57	20
B	1441	4015	79	18

A.1.4　设计

本练习的数据来源于 Briand、Bunse 和 Daly 所做的实验，而后，Briand 等人对该实验进行了进一步的描述［28］。

为了评估在对已有设计进行修改时使用定性的面向对象设计原则的影响，设计了本实验。评估的定性设计原则是由 Coad 和 Yourdon［35］提供的。在实验中，使用两种不同的设计方法分别对两个系统进行设计。其中，“好的”设计方法是指遵循了相关原则的设计方法，而“坏的”设计方法是指没有遵循相关设计原则的设计方法。采用相同的方式将两种设计的布局和内容记录下来，并尽可能使记录的规模相同，即除了在是否遵循原则方面不同之外，采用尽可能相似的方法进行设计。实验目的是评价

在识别出设计中的变化后，质量设计原则能否使变更影响分析更容易。

每个参与者的任务是进行两次独立的变更影响分析，即对每个系统设计各做一次变更影响分析。通过在设计中标记所有必须修改的地方来完成，但并不真正地进行修改。第一个变更影响分析任务是分析客户需求变化的影响；第二个变更影响分析任务是分析系统功能增强的影响。在实验中，收集以下四方面的度量值：

Mod_ Time：识别需要进行修改的地方所花费的时间。

Mod_ Comp：影响分析的完整程度，其具体定义如下：

$$\text{Moc\_ Comp} = \frac{\text{找到的正确的位置数}}{\text{应该找到的总位置数}}$$

Mod_ Corr：影响分析的正确程度，其具体定义如下：

$$\text{Moc\_ Corr} = \frac{\text{找到的正确的位置数}}{\text{标记的总的位置数}}$$

Mod_ Rate：单位时间找到的正确的位置数，即：

$$\text{Moc\_ Rate} = \frac{\text{找到的正确的位置数}}{\text{识别所用的总时间}}$$

为了让每个参与者对好设计和坏设计分别进行影响分析，实验分两次进行。受试者被随机分为两组：A 组 和 B 组。A 组先分析好设计、再分析坏设计；B 组先分析坏设计，再分析好设计。收集的数据见表 A-5。回答以下问题：

204 ~ 210

1. 实验中用到了哪一种实验设计？

2. 定义评估的假设。

3. 如何处理表 A-5 中缺失的数据。

4. 假设使用参数检验方法，评估质量设计原则对四个被测变量的影响。从结果中能够推断出哪些结论？

5. 采用非参数检验方法评估质量设计原则对四个待测变量的影响。从结果中能够推断出哪些结论？与参数检验得出的结论进行比较。

6. 如果使用参数检验是合适的，讨论结果的有效性。

7. 实验的参与者都是学生，并且都是上过软件工程课程的志愿者。那么，实验样本来自于哪个总体？讨论这种类型的取样会对实验的外部有效性产生何种影响？如何进行不同的取样？

表 A-5 设计练习的数据

参与者	组	好的面向对象设计				坏的面向对象设计			
		Mod_Time	Mod_Comp	Mod_Corr	Mod_Rate	Mod_Time	Mod_Comp	Mod_Corr	Mod_Rate
P01	B	—	0.545	0.75	—	—	0.238	0.714	—
P02	B	—	0.818	1	—	—	0.095	1	—
P03	A	20	0.409	1	0.45	25	0.19	1	0.16

（续）

参与者	组	好的面向对象设计				坏的面向对象设计			
		Mod_Time	Mod_Comp	Mod_Corr	Mod_Rate	Mod_Time	Mod_Comp	Mod_Corr	Mod_Rate
P04	B	22	0.818	1	0.818	25	0.238	1	0.2
P05	B	30	0.909	1	0.667	35	0.476	0.909	0.286
P07	A	—	0	—	—	38	0.476	1	0.263
P09	A	—	0.455	1	—	—	0.476	1	—
P10	B	—	0.409	0.9	—	—	0.381	1	—
P11	A	45	0.545	0.923	0.267	50	0.714	1	0.3
P12	B	—	0.773	1	—	—	0.714	1	—
P13	A	40	0.773	1	0.425	40	0.762	1	0.4
P14	B	30	0.909	1	0.667	30	0.333	0.875	0.233
P15	B	—	0.864	1	—	40	0.238	1	0.125
P16	B	30	0.773	1	0.567	—	—	—	—
P17	B	—	0.955	1	—	—	0.286	0.75	—
P18	B	—	0	—	—	—	0.19	1	—
P19	A	29	0.818	1	0.621	27	0.667	1	0.519
P20	A	9	0.591	1	1.444	15	0.19	0.8	0.267
P21	B	20	0.591	1	0.65	35	0.19	1	0.114
P22	B	30	0.682	1	0.5	20	0.714	1	0.75
P23	B	—	0.818	1	—	—	0.476	1	—
P24	A	30	0.773	1	0.567	40	0.762	1	0.4
P25	A	—	0.955	1	—	—	0.667	0.875	—
P26	B	25	0	0	0	25	0.095	0.5	0.08
P27	A	27	0.773	0.944	0.63	36	0.389	0.7	0.194
P28	A	25	0.773	1	0.68	30	0.667	1	0.467
P29	B	44	0.773	1	0.386	23	0.762	1	0.696
P31	A	—	0.409	1	—	—	0.286	0.75	—
P32	A	30	0.909	1	0.667	—	0.5	1	—
P33	A	65	0.818	1	0.277	—	0.619	1	—
P34	A	50	0.636	0.933	0.28	30	0.4	0.889	0.267
P35	A	10	0.591	1	1.3	10	0.667	1	1.4
P36	A	13	1	1	1.692	—	0.619	1	—

A.1.5 审查

本练习参考第13章中的实验案例。

1. 参照第11章中的定义，重写第13章的摘要，使之成为结构化摘要。

2. 针对该实验进行精确复制实验，执行“确定范围”和“计划”步骤。尤其要确定需要多少实验主体参与，才能达到给定的分析置信度。

3. 针对该实验进行差分复制实验，执行“确定范围”步骤。为三种不同的复制处置定义三种目标模板。根据成本、风险和收益讨论每种处置的利弊（见图2-1）。

A.2 回顾

以下列举了一系列问题，它们对研读或者评审描述实验的论文非常重要。使用此问题列表，回顾第12章和13章中描述的案例以及文献中提及的实验。

在读文章时，不仅要将下述问题作为一般问题引导阅读，还要将其作为检查单对照阅读。例如：文章摘要是否很好地描述了论文内容？阅读实验文章需要考虑的方面包括：

- 总体来说，实验可以理解吗？实验有趣吗？
- 该实验有实用价值吗？ 212
- 总结和引用了讨论这个问题的其他实验吗？
- 实验中的实验总体有多少？
- 使用的样本是否具有代表性？
- 是否明确定义了非独立变量和独立变量？
- 是否清晰地陈述了假设？
- 是否明确陈述了实验的设计类型？
- 设计是否正确？
- 操作指南描述是否正确？
- 是否认真分析了实验有效性，是否有说服力？
- 不同类型的有效性威胁处理得恰当吗？
- 数据已经确认吗？
- 统计检验效能足够吗？实验主体足够吗？
- 使用了合适的统计检验么？使用的是参数检验还是非参数检验，且使用正确吗？
- 是否使用了适当的显著性水平？
- 数据解释得正确吗？
- 结论正确吗？
- 结果没有被夸大吗？
- 能够重复这个实验研究吗？

- 数据提供了吗?
- 可能用结果进行元分析吗?
- 阐明了需要进行的进一步工作和实验吗?

A.3 任务

以下任务的总体背景：一个公司希望通过改变软件过程来改进工作方式。假如该公司聘请你作为咨询专家，希望你根据已有过程评估有关的新技术和方法。公司希望知道他们是否应该改变原来的软件过程。

他们希望你找到合适的文献，回顾有关本主题的现有文献，进行实验，并撰写一个包含建议的报告。在建议中，不仅要论述实验的结果，还要讨论与是否改变已有软件过程这一决策相关的问题，以及改变过程的成本、收益等其他相关问题。如果无法确定成本，需要进行成本估算，给出相对成本。

这些任务是特意设计得相对开放的，允许各种解释和讨论。每项任务都对执行该任务所需的前提条件进行了说明，然后简短地对实际的任务进行了描述。应该注意到，以下任务是那些可能进行的实验的一些示例。需要牢记的是，任务的主要目的是提供
213 实践机会，将实验作为评估程序的一部分。

最后，应该指出的是，目前一些组织提供的实验包可以用来做重复实验。实验包使得我们能够在别人的基础上工作，从而也有望通过重复实验获得更普遍的有效结果，因此，实验包是非常重要的。通过互联网可以搜索到一些实验包，也可以通过联系原始实验者获取支持，或许还能得到他们的未公开的实验包。

A.3.1 单元测试和代码审查

公司希望评估引入代码审查是否符合成本效益。目前，针对没有被审查过的代码的单元测试已经完成。这是最好的方式吗?

前提条件

- 一个合适的带有缺陷的程序：这些缺陷能够在审查或测试过程中被找到。
- 一种审查方法：它可以是某种特定的审查方法，但最好是实用的方法，如检查单法。若采用检查单法，则还需要一个检查单。
- 一种测试方法：它也可以是某种特定方法，但最好是基于使用或等价类划分的方法。

任务

- 评估引入代码审查是否符合成本效益。

A.3.2 审查方法

有几种不同的审查方法可用。公司希望从两种候选方法中找出更佳的审查方法。那么对于公司来说，哪种方法更好呢?

前提条件

- 需要提供合适的待审查的软件产品。
- 两种审查方法并附上所需的支持，如，检查单法需要提供检查单、基于视角的阅读法需要提供各种不同阅读视角的描述，参见附录 A.1.5。

任务

- 假设公司希望为已选定的软件产品引入审查方法，应该引入哪种方法呢？确定哪种审查方法在发现缺陷方面的能力更好。好的方法有成本效益吗？ 214

A.3.3 需求表示法

书写需求规格说明是非常重要的，能够使所有读者轻松理解并且理解相同。公司可以从几种不同的表示法中选择，那么描述需求的最好方式是什么？

前提条件

- 以不同表示法书写的一份需求规格说明，例如，自然语言描述和图形法描述的同一份需求规格说明。

任务

- 假设公司目前使用自然语言描述需求规格说明，评估改变公司的需求规格说明表示法是否有益？ 215

附录 B

Experimentation in Software Engineering

统 计 表

本附录包含了显著性水平为5%的统计表。更多详尽表格可以在诸如［119］的统计学书中找到，同时，这些表也能在互联网上找到。本附录的主要目的是提供一些信息，以方便读者理解第10章中描述的假设检验和后面的示例。由于在应用不同的统计检验前了解底层的计算是非常重要的，因此，即使在计算时使用统计包，提供这些信息也是非常重要的。值得一提的是，附录中的表是截取的，例如，对于t-检验，F-检验和Chi-2检验的值可以通过相应的分布来计算得到。

下面的统计表包括：

- t-检验（见10.3.4节、10.3.7节和表B-1）
- Chi-2检验（见10.3.12节和表B-2）
- Mann-Whitney检验（见10.3.5节和表B-3）
- Wilcoxon检验（见10.3.8节和表B-4）
- F-检验（见10.3.6节、10.3.10节和表B-5）

请注意，在表B-3中，N_A代表小样本的量，N_B代表大样本的量。

请注意，表B-5提供了显著性水平为0.025 %时的F分布，其中f_1、f_2代表自由度，相当于$F_{0.0025,f_1,f_2}$。

表B-1 双侧t-检验临界值表（5%），参见10.3.4和10.3.7节

自 由 度	t-值	自 由 度	t-值
1	12.706	14	2.145
2	4.303	15	2.131
3	3.182	16	2.120
4	2.776	17	2.110
5	2.571	18	2.101
6	2.447	19	2.093
7	2.365	20	2.086
8	2.306	21	2.080
9	2.262	22	2.074
10	2.228	23	2.069
11	2.201	24	2.064
12	2.179	25	2.060
13	2.160	26	2.056

（续）

自 由 度	t-值	自 由 度	t-值
27	2.052	40	2.021
28	2.048	60	2.000
29	2.045	120	1.980
30	2.042	∞	1.960

表 B-2 单侧 Chi-2 检验临界值表（5%），参见 10.3.12 节

自 由 度	χ^2	自 由 度	χ^2
1	3.84	18	28.87
2	5.99	19	30.14
3	7.81	20	31.41
4	9.49	21	32.67
5	11.07	22	33.92
6	12.59	23	35.17
7	14.07	24	36.42
8	15.51	25	37.65
9	16.92	26	38.88
10	18.31	27	40.11
11	19.68	28	41.34
12	21.03	29	42.56
13	22.36	30	43.77
14	23.68	40	55.76
15	25.00	60	79.08
16	26.30	80	101.88
17	27.59	100	124.34

表 B-3 双侧 Mann-Whitney 检验临界值表（5%），参见 10.3.5 节

N_A \ N_B	5	6	7	8	9	10	11	12
3	0	1	1	2	2	3	3	4
4	1	2	3	4	4	5	6	7
5	2	3	5	6	7	8	9	11
6		5	6	8	10	11	13	14

（续）

N_A \ N_B	5	6	7	8	9	10	11	12
7			8	10	12	14	16	18
8				13	15	17	19	22
9					17	20	23	26
10						23	26	29
11							30	33
12								37

表 B-4 双侧配对 Wilcoxon 检验临界值表（5%），参见 10.3.8 节

n	T
6	0
7	2
8	3
9	5
10	8
11	10
12	13
13	17
14	21
15	25
16	29
17	34
18	40
19	46
20	52
22	66
25	89

表 B-5 双侧 F 检验的临界值表(5%),参见 10.3.6 节。对 ANOVA 检验,本表视同显著性水平 2.5% 的临界值表,参见 10.3.10 节

f_1 / f_2	1	2	3	4	5	6	7	8	9	10	12	15	20	30	40	60	120	oo
1	648	800	864	900	922	937	948	957	963	969	977	985	993	1 001	1 006	1 010	1 014	1 018
2	38.5	39.0	39.2	39.2	39.3	39.3	39.4	39.4	39.4	39.4	39.4	39.4	39.4	39.5	39.5	39.5	39.5	39.5
3	17.4	16.0	15.4	15.1	14.9	14.7	14.6	14.5	14.5	14.4	14.3	14.2	14.2	14.1	14.0	14.0	14.0	13.9
4	12.2	10.6	9.98	9.60	9.36	9.20	9.07	8.98	8.90	8.84	8.75	8.66	8.56	8.46	8.41	8.36	8.31	8.26
5	10.0	8.43	7.76	7.39	7.15	6.98	6.85	6.76	6.68	6.62	6.52	6.43	6.33	6.23	6.18	6.12	6.07	6.02
6	8.81	7.26	6.60	6.23	5.99	5.82	5.70	5.60	5.52	5.46	5.37	5.27	5.17	5.07	5.01	4.96	4.90	4.85
7	8.07	6.54	5.89	5.52	5.29	5.12	4.99	4.90	4.82	4.76	4.67	4.57	4.47	4.36	4.31	4.25	4.20	4.14
8	7.57	6.06	5.42	5.05	4.82	4.65	4.53	4.43	4.36	4.30	4.20	4.10	4.00	3.89	3.84	3.78	3.73	3.67
9	7.21	5.71	5.08	4.72	4.48	4.32	4.20	4.10	4.03	3.96	3.87	3.77	3.67	3.56	3.51	3.45	3.39	3.33
10	6.94	5.46	4.83	4.47	4.24	4.07	3.95	3.85	3.78	3.72	3.62	3.52	3.42	3.31	3.26	3.20	3.14	3.08
12	6.55	5.10	4.47	4.12	3.89	3.73	3.61	3.51	3.44	3.37	3.28	3.18	3.07	2.96	2.91	2.85	2.79	2.72
15	6.20	4.76	4.15	3.80	3.58	3.41	3.29	3.20	3.12	3.06	2.96	2.86	2.76	2.64	2.59	2.52	2.46	2.40
20	5.87	4.46	3.86	3.51	3.29	3.13	3.01	2.91	2.84	2.77	2.68	2.57	2.46	2.35	2.29	2.22	2.16	2.09
30	5.57	4.18	3.59	3.25	3.03	2.87	2.75	2.65	2.57	2.51	2.41	2.31	2.20	2.07	2.01	1.94	1.87	1.79
40	5.42	4.05	3.46	3.13	2.90	2.74	2.62	2.53	2.45	2.39	2.29	2.18	2.07	1.94	1.88	1.80	1.72	1.64
60	5.29	3.93	3.34	3.01	2.79	2.63	2.51	2.41	2.33	2.27	2.17	2.06	1.94	1.82	1.74	1.67	1.58	1.48
120	5.15	3.80	3.23	2.89	2.67	2.52	2.39	2.30	2.22	2.16	2.05	1.94	1.82	1.69	1.61	1.53	1.43	1.31
∞	5.02	3.69	3.12	2.79	2.57	2.41	2.29	2.19	2.11	2.05	1.94	1.83	1.71	1.57	1.48	1.39	1.27	1.00

参考文献

1. Anastas, J.W., MacDonald, M.L.: Research Design for the Social Work and the Human Services, 2nd edn. Columbia University Press, New York (2000)
2. Andersson, C., Runeson, P.: A spiral process model for case studies on software quality monitoring – method and metrics. Softw. Process: Improv. Pract. **12**(2), 125–140 (2007). doi: 10.1002/spip.311
3. Andrews, A.A., Pradhan, A.S.: Ethical issues in empirical software engineering: the limits of policy. Empir. Softw. Eng. **6**(2), 105–110 (2001)
4. American Psychological Association: Ethical principles of psychologists and code of conduct. Am. Psychol. **47**, 1597–1611 (1992)
5. Avison, D., Baskerville, R., Myers, M.: Controlling action research projects. Inf. Technol. People **14**(1), 28–45 (2001). doi: 10.1108/09593840110384762. URL http://www.emeraldinsight.com/10.1108/09593840110384762
6. Babbie, E.R.: Survey Research Methods. Wadsworth, Belmont (1990)
7. Basili, V.R.: Quantitative evaluation of software engineering methodology. In: Proceedings of the First Pan Pacific Computer Conference, vol. 1, pp. 379–398. Australian Computer Society, Melbourne (1985)
8. Basili, V.R.: Software development: a paradigm for the future. In: Proceedings of the 13th Annual International Computer Software and Applications Conference, COMPSAC'89, Orlando, pp. 471–485. IEEE Computer Society Press, Washington (1989)
9. Basili, V.R.: The experimental paradigm in software engineering. In: H.D. Rombach, V.R. Basili, R.W. Selby (eds.) Experimental Software Engineering Issues: Critical Assessment and Future Directives. Lecture Notes in Computer Science, vol. 706. Springer, Berlin Heidelberg (1993)
10. Basili, V.R.: Evolving and packaging reading technologies. J. Syst. Softw. **38**(1), 3–12 (1997)
11. Basili, V.R., Weiss, D.M.: A methodology for collecting valid software engineering data. IEEE Trans. Softw. Eng. **10**(6), 728–737 (1984)
12. Basili, V.R., Selby, R.W.: Comparing the effectiveness of software testing strategies. IEEE Trans. Softw. Eng. **13**(12), 1278–1298 (1987)
13. Basili, V.R., Rombach, H.D.: The TAME project: towards improvement-oriented software environments. IEEE Trans. Softw. Eng. **14**(6), 758–773 (1988)
14. Basili, V.R., Green, S.: Software process evaluation at the SEL. IEEE Softw. **11**(4), pp. 58–66 (1994)
15. Basili, V.R., Selby, R.W., Hutchens, D.H.: Experimentation in software engineering. IEEE Trans. Softw. Eng. **12**(7), 733–743 (1986)
16. Basili, V.R., Caldiera, G., Rombach, H.D.: Experience factory. In: J.J. Marciniak (ed.) Encyclopedia of Software Engineering, pp. 469–476. Wiley, New York (1994)

17. Basili, V.R., Caldiera, G., Rombach, H.D.: Goal Question Metrics paradigm. In: J.J. Marciniak (ed.) Encyclopedia of Software Engineering, pp. 528–532. Wiley (1994)
18. Basili, V.R., Green, S., Laitenberger, O., Lanubile, F., Shull, F., Sørumgård, S., Zelkowitz, M.V.: The empirical investigation of perspective-based reading. Empir. Soft. Eng. **1**(2), 133–164 (1996)
19. Basili, V.R., Green, S., Laitenberger, O., Lanubile, F., Shull, F., Sørumgård, S., Zelkowitz, M.V.: Lab package for the empirical investigation of perspective-based reading. Technical report, Univeristy of Maryland (1998). URL http://www.cs.umd.edu/projects/SoftEng/ESEG/manual/pbr_package/manual.html
20. Basili, V.R., Shull, F., Lanubile, F.: Building knowledge through families of experiments. IEEE Trans. Softw. Eng. **25**(4), 456–473 (1999)
21. Baskerville, R.L., Wood-Harper, A.T.: A critical perspective on action research as a method for information systems research. J. Inf. Technol. **11**(3), 235–246 (1996). doi: 10.1080/026839696345289
22. Benbasat, I., Goldstein, D.K., Mead, M.: The case research strategy in studies of information systems. MIS Q. **11**(3), 369 (1987). doi: 10.2307/248684
23. Bergman, B., Klefsjö, B.: Quality from Customer Needs to Customer Satisfaction. Studentlitteratur, Lund (2010)
24. Brereton, P., Kitchenham, B.A., Budgen, D., Turner, M., Khalil, M.: Lessons from applying the systematic literature review process within the software engineering domain. J. Syst. Softw. **80**(4), 571–583 (2007). doi: 10.1016/j.jss.2006.07.009
25. Brereton, P., Kitchenham, B.A., Budgen, D.: Using a protocol template for case study planning. In: Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering. University of Bari, Italy (2008)
26. Briand, L.C., Differding, C.M., Rombach, H.D.: Practical guidelines for measurement-based process improvement. Softw. Process: Improv. Pract. **2**(4), 253–280 (1996)
27. Briand, L.C., El Emam, K., Morasca, S.: On the application of measurement theory in software engineering. Empir. Softw. Eng. **1**(1), 61–88 (1996)
28. Briand, L.C., Bunse, C., Daly, J.W.: A controlled experiment for evaluating quality guidelines on the maintainability of object-oriented designs. IEEE Trans. Softw. Eng. **27**(6), 513–530 (2001)
29. British Psychological Society: Ethical principles for conducting research with human participants. Psychologist **6**(1), 33–35 (1993)
30. Budgen, D., Kitchenham, B.A., Charters, S., Turner, M., Brereton, P., Linkman, S.: Presenting software engineering results using structured abstracts: a randomised experiment. Empir. Softw. Eng. **13**, 435–468 (2008). doi: 10.1007/s10664-008-9075-7
31. Budgen, D., Burn, A.J., Kitchenham, B.A.: Reporting computing projects through structured abstracts: a quasi-experiment. Empir. Softw. Eng. **16**(2), 244–277 (2011). doi: 10.1007/s10664-010-9139-3
32. Campbell, D.T., Stanley, J.C.: Experimental and Quasi-experimental Designs for Research. Houghton Mifflin Company, Boston (1963)
33. Chrissis, M.B., Konrad, M., Shrum, S.: CMMI(R): Guidelines for process integration and product improvement. Technical report, SEI (2003)
34. Ciolkowski, M., Differding, C.M., Laitenberger, O., Münch, J.: Empirical investigation of perspective-based reading: A replicated experiment. Technical report, 97-13, ISERN (1997)
35. Coad, P., Yourdon, E.: Object-Oriented Design, 1st edn. Prentice-Hall, Englewood (1991)
36. Cohen, J.: Weighted kappa: nominal scale agreement with provision for scaled disagreement or partial credit. Psychol. Bull. **70**, 213–220 (1968)
37. Cook, T.D., Campbell, D.T.: Quasi-experimentation – Design and Analysis Issues for Field Settings. Houghton Mifflin Company, Boston (1979)
38. Corbin, J., Strauss, A.: Basics of Qualitative Research, 3rd edn. SAGE, Los Angeles (2008)
39. Cruzes, D.S., Dybå, T.: Research synthesis in software engineering: a tertiary study. Inf. Softw. Technol. **53**(5), 440–455 (2011). doi: 10.1016/j.infsof.2011.01.004

40. Dalkey, N., Helmer, O.: An experimental application of the delphi method to the use of experts. Manag. Sci. **9**(3), 458–467 (1963)
41. DeMarco, T.: Controlling Software Projects. Yourdon Press, New York (1982)
42. Demming, W.E.: Out of the Crisis. MIT Centre for Advanced Engineering Study, MIT Press, Cambridge, MA (1986)
43. Dieste, O., Grimán, A., Juristo, N.: Developing search strategies for detecting relevant experiments. Empir. Softw. Eng. **14**, 513–539 (2009). URL http://dx.doi.org/10.1007/s10664-008-9091-7
44. Dittrich, Y., Rönkkö, K., Eriksson, J., Hansson, C., Lindeberg, O.: Cooperative method development. Empir. Softw. Eng. **13**(3), 231–260 (2007). doi: 10.1007/s10664-007-9057-1
45. Doolan, E.P.: Experiences with Fagan's inspection method. Softw. Pract. Exp. **22**(2), 173–182 (1992)
46. Dybå, T., Dingsøyr, T.: Empirical studies of agile software development: a systematic review. Inf. Softw. Technol. **50**(9-10), 833–859 (2008). doi: DOI:10.1016/j.infsof.2008.01.006
47. Dybå, T., Dingsøyr, T.: Strength of evidence in systematic reviews in software engineering. In: Proceedings of the 2nd ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM '08, Kaiserslautern, pp. 178–187. ACM, New York (2008). doi: http://doi.acm.org/10.1145/1414004.1414034
48. Dybå, T., Kitchenham, B.A., Jørgensen, M.: Evidence-based software engineering for practitioners. IEEE Softw. **22**, 58–65 (2005). doi: http://doi.ieeecomputersociety.org/10.1109/MS.2005.6
49. Dybå, T., Kampenes, V.B., Sjøberg, D.I.K.: A systematic review of statistical power in software engineering experiments. Inf. Softw. Technol. **48**(8), 745–755 (2006). doi: 10.1016/j.infsof.2005.08.009
50. Easterbrook, S., Singer, J., Storey, M.-A., Damian, D.: Selecting empirical methods for software engineering research. In: F. Shull, J. Singer, D.I. Sjøberg (eds.) Guide to Advanced Empirical Software Engineering. Springer, London (2008)
51. Eick, S.G., Loader, C.R., Long, M.D., Votta, L.G., Vander Wiel, S.A.: Estimating software fault content before coding. In: Proceedings of the 14th International Conference on Software Engineering, Melbourne, pp. 59–65. ACM Press, New York (1992)
52. Eisenhardt, K.M.: Building theories from case study research. Acad. Manag. Rev. **14**(4), 532 (1989). doi: 10.2307/258557
53. Endres, A., Rombach, H.D.: A Handbook of Software and Systems Engineering – Empirical Observations, Laws and Theories. Pearson Addison-Wesley, Harlow/New York (2003)
54. Fagan, M.E.: Design and code inspections to reduce errors in program development. IBM Syst. J. **15**(3), 182–211 (1976)
55. Fenton, N.: Software measurement: A necessary scientific basis. IEEE Trans. Softw. Eng. **3**(20), 199–206 (1994)
56. Fenton, N., Pfleeger, S.L.: Software Metrics: A Rigorous and Practical Approach, 2nd edn. International Thomson Computer Press, London (1996)
57. Fenton, N., Pfleeger, S.L., Glass, R.: Science and substance: A challenge to software engineers. IEEE Softw. **11**, 86–95 (1994)
58. Fink, A.: The Survey Handbook, 2nd edn. SAGE, Thousand Oaks/London (2003)
59. Flyvbjerg, B.: Five misunderstandings about case-study research. In: Qualitative Research Practice, concise paperback edn., pp. 390–404. SAGE, London (2007)
60. Frigge, M., Hoaglin, D.C., Iglewicz, B.: Some implementations of the boxplot. Am. Stat. **43**(1), 50–54 (1989)
61. Fusaro, P., Lanubile, F., Visaggio, G.: A replicated experiment to assess requirements inspection techniques. Empir. Softw. Eng. **2**(1), 39–57 (1997)
62. Glass, R.L.: The software research crisis. IEEE Softw. **11**, 42–47 (1994)
63. Glass, R.L., Vessey, I., Ramesh, V.: Research in software engineering: An analysis of the literature. Inf. Softw. Technol. **44**(8), 491–506 (2002). doi: 10.1016/S0950-5849(02)00049-6

64. Gómez, O.S., Juristo, N., Vegas, S.: Replication types in experimental disciplines. In: Proceedings of the 4th ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, Bolzano-Bozen (2010)
65. Gorschek, T., Wohlin, C.: Requirements abstraction model. Requir. Eng. **11**, 79–101 (2006). doi: 10.1007/s00766-005-0020-7
66. Gorschek, T., Garre, P., Larsson, S., Wohlin, C.: A model for technology transfer in practice. IEEE Softw. **23**(6), 88–95 (2006)
67. Gorschek, T., Garre, P., Larsson, S., Wohlin, C.: Industry evaluation of the requirements abstraction model. Requir. Eng. **12**, 163–190 (2007). doi: 10.1007/s00766-007-0047-z
68. Grady, R.B., Caswell, D.L.: Software Metrics: Establishing a Company-Wide Program. Prentice-Hall, Englewood (1994)
69. Grant, E.E., Sackman, H.: An exploratory investigation of programmer performance under on-line and off-line conditions. IEEE Trans. Human Factor Electron. **HFE-8**(1), 33–48 (1967)
70. Gregor, S.: The nature of theory in information systems. MIS Q. **30**(3), 491–506 (2006)
71. Hall, T., Flynn, V.: Ethical issues in software engineering research: a survey of current practice. Empir. Softw. Eng. **6**, 305–317 (2001)
72. Hannay, J.E., Sjøberg, D.I.K., Dybå, T.: A systematic review of theory use in software engineering experiments. IEEE Trans. Softw. Eng. **33**(2), 87–107 (2007). doi: 10.1109/TSE.2007.12
73. Hannay, J.E., Dybå, T., Arisholm, E., Sjøberg, D.I.K.: The effectiveness of pair programming: a meta-analysis. Inf. Softw. Technol. **51**(7), 1110–1122 (2009). doi: 10.1016/j.infsof.2009.02.001
74. Hayes, W.: Research synthesis in software engineering: a case for meta-analysis. In: Proceedings of the 6th International Software Metrics Symposium, Boca Raton, pp. 143–151 (1999)
75. Hetzel, B.: Making Software Measurement Work: Building an Effective Measurement Program. Wiley, New York (1993)
76. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. MIS Q. **28**(1), 75–105 (2004)
77. Höst, M., Regnell, B., Wohlin, C.: Using students as subjects – a comparative study of students and professionals in lead-time impact assessment. Empir. Softw. Eng. **5**(3), 201–214 (2000)
78. Höst, M., Wohlin, C., Thelin, T.: Experimental context classification: Incentives and experience of subjects. In: Proceedings of the 27th International Conference on Software Engineering, St. Louis, pp. 470–478 (2005)
79. Höst, M., Runeson, P.: Checklists for software engineering case study research. In: Proceedings of the 1st International Symposium on Empirical Software Engineering and Measurement, Madrid, pp. 479–481 (2007)
80. Hove, S.E., Anda, B.: Experiences from conducting semi-structured interviews in empirical software engineering research. In: Proceedings of the 11th IEEE International Software Metrics Symposium, pp. 1–10. IEEE Computer Society Press, Los Alamitos (2005)
81. Humphrey, W.S.: Managing the Software Process. Addison-Wesley, Reading (1989)
82. Humphrey, W.S.: A Discipline for Software Engineering. Addison Wesley, Reading (1995)
83. Humphrey, W.S.: Introduction to the Personal Software Process. Addison Wesley, Reading (1997)
84. IEEE: IEEE standard glossary of software engineering terminology. Technical Report, IEEE Std 610.12-1990, IEEE (1990)
85. Iversen, J.H., Mathiassen, L., Nielsen, P.A.: Managing risk in software process improvement: an action research approach. MIS Q. **28**(3), 395–433 (2004)
86. Jedlitschka, A., Pfahl, D.: Reporting guidelines for controlled experiments in software engineering. In: Proceedings of the 4th International Symposium on Empirical Software Engineering, Noosa Heads, pp. 95–104 (2005)
87. Johnson, P.M., Tjahjono, D.: Does every inspection really need a meeting? Empir. Softw. Eng. **3**(1), 9–35 (1998)

88. Juristo, N., Moreno, A.M.: Basics of Software Engineering Experimentation. Springer, Kluwer Academic Publishers, Boston (2001)
89. Juristo, N., Vegas, S.: The role of non-exact replications in software engineering experiments. Empir. Softw. Eng. **16**, 295–324 (2011). doi: 10.1007/s10664-010-9141-9
90. Kachigan, S.K.: Statistical Analysis: An Interdisciplinary Introduction to Univariate and Multivariate Methods. Radius Press, New York (1986)
91. Kachigan, S.K.: Multivariate Statistical Analysis: A Conceptual Introduction, 2nd edn. Radius Press, New York (1991)
92. Kampenes, V.B., Dyba, T., Hannay, J.E., Sjø berg, D.I.K.: A systematic review of effect size in software engineering experiments. Inf. Softw. Technol. **49**(11–12), 1073–1086 (2007). doi: 10.1016/j.infsof.2007.02.015
93. Karahasanović, A., Anda, B., Arisholm, E., Hove, S.E., Jørgensen, M., Sjøberg, D., Welland, R.: Collecting feedback during software engineering experiments. Empir. Softw. Eng. **10**(2), 113–147 (2005). doi: 10.1007/s10664-004-6189-4. URL http://www.springerlink.com/index/10.1007/s10664-004-6189-4
94. Karlström, D., Runeson, P., Wohlin, C.: Aggregating viewpoints for strategic software process improvement. IEE Proc. Softw. **149**(5), 143–152 (2002). doi: 10.1049/ip-sen:20020696
95. Kitchenham, B.A.: The role of replications in empirical software engineering – a word of warning. Empir. Softw. Eng. **13**, 219–221 (2008). URL 10.1007/s10664-008-9061-0
96. Kitchenham, B.A., Charters, S.: Guidelines for performing systematic literature reviews in software engineering (version 2.3). Technical Report, EBSE Technical Report EBSE-2007-01, Keele University and Durham University (2007)
97. Kitchenham, B.A., Pickard, L.M., Pfleeger, S.L.: Case studies for method and tool evaluation. IEEE Softw. **12**(4), 52–62 (1995)
98. Kitchenham, B.A., Pfleeger, S.L., Pickard, L.M., Jones, P.W., Hoaglin, D.C., El Emam, K., Rosenberg, J.: Preliminary guidelines for empirical research in software engineering. IEEE Trans. Softw. Eng. **28**(8), 721–734 (2002). doi: 10.1109/TSE.2002.1027796. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1027796
99. Kitchenham, B., Fry, J., Linkman, S.G.: The case against cross-over designs in software engineering. In: Proceedings of the 11th International Workshop on Software Technology and Engineering Practice, Amsterdam, pp. 65–67. IEEE Computer Society, Los Alamitos (2003)
100. Kitchenham, B.A., Dybå, T., Jørgensen, M.: Evidence-based software engineering. In: Proceedings of the 26th International Conference on Software Engineering, Edinburgh, pp. 273–281 (2004)
101. Kitchenham, B.A., Al-Khilidar, H., Babar, M.A., Berry, M., Cox, K., Keung, J., Kurniawati, F., Staples, M., Zhang, H., Zhu, L.: Evaluating guidelines for reporting empirical software engineering studies. Empir. Softw. Eng. **13**(1), 97–121 (2007). doi: 10.1007/s10664-007-9053-5. URL http://www.springerlink.com/index/10.1007/s10664-007-9053-5
102. Kitchenham, B.A., Jeffery, D.R., Connaughton, C.: Misleading metrics and unsound analyses. IEEE Softw. **24**, 73–78 (2007). doi: 10.1109/MS.2007.49
103. Kitchenham, B.A., Brereton, P., Budgen, D., Turner, M., Bailey, J., Linkman, S.G.: Systematic literature reviews in software engineering – a systematic literature review. Inf. Softw. Technol. **51**(1), 7–15 (2009). doi: 10.1016/j.infsof.2008.09.009. URL http://www.dx.doi.org/10.1016/j.infsof.2008.09.009
104. Kitchenham, B.A., Pretorius, R., Budgen, D., Brereton, P., Turner, M., Niazi, M., Linkman, S.: Systematic literature reviews in software engineering – a tertiary study. Inf. Softw. Technol. **52**(8), 792–805 (2010). doi: 10.1016/j.infsof.2010.03.006
105. Kitchenham, B.A., Sjøberg, D.I.K., Brereton, P., Budgen, D., Dybå, T., Höst, M., Pfahl, D., Runeson, P.: Can we evaluate the quality of software engineering experiments? In: Proceedings of the 4th ACM-IEEE International Symposium on Empirical Software Engineering and Measurement. ACM, Bolzano/Bozen (2010)
106. Kitchenham, B.A., Budgen, D., Brereton, P.: Using mapping studies as the basis for further research – a participant-observer case study. Inf. Softw. Technol. **53**(6), 638–651 (2011). doi: 10.1016/j.infsof.2010.12.011

107. Laitenberger, O., Atkinson, C., Schlich, M., El Emam, K.: An experimental comparison of reading techniques for defect detection in UML design documents. J. Syst. Softw. **53**(2), 183–204 (2000)
108. Larsson, R.: Case survey methodology: quantitative analysis of patterns across case studies. Acad. Manag. J. **36**(6), 1515–1546 (1993)
109. Lee, A.S.: A scientific methodology for MIS case studies. MIS Q. **13**(1), 33 (1989). doi: 10.2307/248698. URL http://www.jstor.org/stable/248698?origin=crossref
110. Lehman, M.M.: Program, life-cycles and the laws of software evolution. Proc. IEEE **68**(9), 1060–1076 (1980)
111. Lethbridge, T.C., Sim, S.E., Singer, J.: Studying software engineers: data collection techniques for software field studies. Empir. Softw. Eng. **10**, 311–341 (2005)
112. Linger, R.: Cleanroom process model. IEEE Softw. pp. 50–58 (1994)
113. Linkman, S., Rombach, H.D.: Experimentation as a vehicle for software technology transfer – a family of software reading techniques. Inf. Softw. Technol. **39**(11), 777–780 (1997)
114. Lucas, W.A.: The case survey method: aggregating case experience. Technical Report, R-1515-RC, The RAND Corporation, Santa Monica (1974)
115. Lucas, H.C., Kaplan, R.B.: A structured programming experiment. Comput. J. **19**(2), 136–138 (1976)
116. Lyu, M.R. (ed.): Handbook of Software Reliability Engineering. McGraw-Hill, New York (1996)
117. Maldonado, J.C., Carver, J., Shull, F., Fabbri, S., Dória, E., Martimiano, L., Mendonça, M., Basili, V.: Perspective-based reading: a replicated experiment focused on individual reviewer effectiveness. Empir. Softw. Eng. **11**, 119–142 (2006). doi: 10.1007/s10664-006-5967-6
118. Manly, B.F.J.: Multivariate Statistical Methods: A Primer, 2nd edn. Chapman and Hall, London (1994)
119. Marascuilo, L.A., Serlin, R.C.: Statistical Methods for the Social and Behavioral Sciences. W. H. Freeman and Company, New York (1988)
120. Miller, J.: Estimating the number of remaining defects after inspection. Softw. Test. Verif. Reliab. **9**(4), 167–189 (1999)
121. Miller, J.: Applying meta-analytical procedures to software engineering experiments. J. Syst. Softw. **54**(1), 29–39 (2000)
122. Miller, J.: Statistical significance testing: a panacea for software technology experiments? J. Syst. Softw. **73**, 183–192 (2004). doi: http://dx.doi.org/10.1016/j.jss.2003.12.019
123. Miller, J.: Replicating software engineering experiments: a poisoned chalice or the holy grail. Inf. Softw. Technol. **47**(4), 233–244 (2005)
124. Miller, J., Wood, M., Roper, M.: Further experiences with scenarios and checklists. Empir. Softw. Eng. **3**(1), 37–64 (1998)
125. Montgomery, D.C.: Design and Analysis of Experiments, 5th edn. Wiley, New York (2000)
126. Myers, G.J.: A controlled experiment in program testing and code walkthroughs/inspections. Commun. ACM **21**, 760–768 (1978). doi: http://doi.acm.org/10.1145/359588.359602
127. Noblit, G.W., Hare, R.D.: Meta-Ethnography: Synthesizing Qualitative Studies. Sage Publications, Newbury Park (1988)
128. Ohlsson, M.C., Wohlin, C.: A project effort estimation study. Inf. Softw. Technol. **40**(14), 831–839 (1998)
129. Owen, S., Brereton, P., Budgen, D.: Protocol analysis: a neglected practice. Commun. ACM **49**(2), 117–122 (2006). doi: 10.1145/1113034.1113039
130. Paulk, M.C., Curtis, B., Chrissis, M.B., Weber, C.V.: Capability maturity model for software. Technical Report, CMU/SEI-93-TR-24, Software Engineering Institute, Pittsburgh (1993)
131. Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M.: Systematic mapping studies in software engineering. In: Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering, Electronic Workshops in Computing (eWIC). BCS, University of Bari, Italy (2008)
132. Petersen, K., Wohlin, C.: Context in industrial software engineering research. In: Proceedings of the 3rd ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, Lake Buena Vista, pp. 401–404 (2009)

133. Pfleeger, S.L.: Experimental design and analysis in software engineering part 1–5. ACM Sigsoft, Softw. Eng. Notes, **19**(4), 16–20; **20**(1), 22–26; **20**(2), 14–16; **20**(3), 13–15; **20**, (1994)
134. Pfleeger, S.L., Atlee, J.M.: Software Engineering: Theory and Practice, 4th edn. Pearson Prentice-Hall, Upper Saddle River (2009)
135. Pickard, L.M., Kitchenham, B.A., Jones, P.W.: Combining empirical results in software engineering. Inf. Softw. Technol. **40**(14), 811–821 (1998). doi: 10.1016/S0950-5849(98)00101-3
136. Porter, A.A., Votta, L.G.: An experiment to assess different defect detection methods for software requirements inspections. In: Proceedings of the 16th International Conference on Software Engineering, Sorrento, pp. 103–112 (1994)
137. Porter, A.A., Votta, L.G.: Comparing detection methods for software requirements inspection: a replicated experiment. IEEE Trans. Softw. Eng. **21**(6), 563–575 (1995)
138. Porter, A.A., Votta, L.G.: Comparing detection methods for software requirements inspection: a replicated experimentation: a replication using professional subjects. Empir. Softw. Eng. **3**(4), 355–380 (1998)
139. Porter, A.A., Siy, H.P., Toman, C.A., Votta, L.G.: An experiment to assess the cost-benefits of code inspections in large scale software development. IEEE Trans. Softw. Eng. **23**(6), 329–346 (1997)
140. Potts, C.: Software engineering research revisited. IEEE Softw. pp. 19–28 (1993)
141. Rainer, A.W.: The longitudinal, chronological case study research strategy: a definition, and an example from IBM Hursley Park. Inf. Softw. Technol. **53**(7), 730–746 (2011)
142. Robinson, H., Segal, J., Sharp, H.: Ethnographically-informed empirical studies of software practice. Inf. Softw. Technol. **49**(6), 540–551 (2007). doi: 10.1016/j.infsof.2007.02.007
143. Robson, C.: Real World Research: A Resource for Social Scientists and Practitioners-Researchers, 1st edn. Blackwell, Oxford/Cambridge (1993)
144. Robson, C.: Real World Research: A Resource for Social Scientists and Practitioners-Researchers, 2nd edn. Blackwell, Oxford/Madden (2002)
145. Runeson, P., Skoglund, M.: Reference-based search strategies in systematic reviews. In: Proceedings of the 13th International Conference on Empirical Assessment and Evaluation in Software Engineering. Electronic Workshops in Computing (eWIC). BCS, Durham University, UK (2009)
146. Runeson, P., Höst, M., Rainer, A.W., Regnell, B.: Case Study Research in Software Engineering. Guidelines and Examples. Wiley, Hoboken (2012)
147. Sandahl, K., Blomkvist, O., Karlsson, J., Krysander, C., Lindvall, M., Ohlsson, N.: An extended replication of an experiment for assessing methods for software requirements. Empir. Softw. Eng. **3**(4), 381–406 (1998)
148. Seaman, C.B.: Qualitative methods in empirical studies of software engineering. IEEE Trans. Softw. Eng. **25**(4), 557–572 (1999)
149. Selby, R.W., Basili, V.R., Baker, F.T.: Cleanroom software development: An empirical evaluation. IEEE Trans. Softw. Eng. **13**(9), 1027–1037 (1987)
150. Shepperd, M.: Foundations of Software Measurement. Prentice-Hall, London/New York (1995)
151. Shneiderman, B., Mayer, R., McKay, D., Heller, P.: Experimental investigations of the utility of detailed flowcharts in programming. Commun. ACM **20**, 373–381 (1977). doi: 10.1145/359605.359610
152. Shull, F.: Developing techniques for using software documents: a series of empirical studies. Ph.D. thesis, Computer Science Department, University of Maryland, USA (1998)
153. Shull, F., Basili, V.R., Carver, J., Maldonado, J.C., Travassos, G.H., Mendonça, M.G., Fabbri, S.: Replicating software engineering experiments: addressing the tacit knowledge problem. In: Proceedings of the 1st International Symposium on Empirical Software Engineering, Nara, pp. 7–16 (2002)
154. Shull, F., Mendoncça, M.G., Basili, V.R., Carver, J., Maldonado, J.C., Fabbri, S., Travassos, G.H., Ferreira, M.C.: Knowledge-sharing issues in experimental software engineering. Empir. Softw. Eng. **9**, 111–137 (2004). doi: 10.1023/B:EMSE.0000013516.80487.33

155. Shull, F., Carver, J., Vegas, S., Juristo, N.: The role of replications in empirical software engineering. Empir. Softw. Eng. **13**, 211–218 (2008). doi: 10.1007/s10664-008-9060-1
156. Sieber, J.E.: Protecting research subjects, employees and researchers: implications for software engineering. Empir. Softw. Eng. **6**(4), 329–341 (2001)
157. Siegel, S., Castellan, J.: Nonparametric Statistics for the Behavioral Sciences, 2nd edn. McGraw-Hill International Editions, New York (1988)
158. Singer, J., Vinson, N.G.: Why and how research ethics matters to you. Yes, you! Empir. Softw. Eng. **6**, 287–290 (2001). doi: 10.1023/A:1011998412776
159. Singer, J., Vinson, N.G.: Ethical issues in empirical studies of software engineering. IEEE Trans. Softw. Eng. **28**(12), 1171–1180 (2002). doi: 10.1109/TSE.2002.1158289. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1158289
160. Simon S.: Fermat's Last Theorem. Fourth Estate, London (1997)
161. Sjøberg, D.I.K., Hannay, J.E., Hansen, O., Kampenes, V.B., Karahasanovic, A., Liborg, N.-K., Rekdal, A.C.: A survey of controlled experiments in software engineering. IEEE Trans. Softw. Eng. **31**(9), 733–753 (2005). doi: 10.1109/TSE.2005.97. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1514443
162. Sjøberg, D.I.K., Dybå, T., Anda, B., Hannay, J.E.: Building theories in software engineering. In: Shull, F., Singer, J., Sjøberg D. (eds.) Guide to Advanced Empirical Software Engineering. Springer, London (2008)
163. Sommerville, I.: Software Engineering, 9th edn. Addison-Wesley, Wokingham, England/Reading (2010)
164. Sørumgård, S.: Verification of process conformance in empirical studies of software development. Ph.D. thesis, The Norwegian University of Science and Technology, Department of Computer and Information Science, Norway (1997)
165. Stake, R.E.: The Art of Case Study Research. SAGE Publications, Thousand Oaks (1995)
166. Staples, M., Niazi, M.: Experiences using systematic review guidelines. J. Syst. Softw. **80**(9), 1425–1437 (2007). doi: 10.1016/j.jss.2006.09.046
167. Thelin, T., Runeson, P.: Capture-recapture estimations for perspective-based reading – a simulated experiment. In: Proceedings of the 1st International Conference on Product Focused Software Process Improvement (PROFES), Oulu, pp. 182–200 (1999)
168. Thelin, T., Runeson, P., Wohlin, C.: An experimental comparison of usage-based and checklist-based reading. IEEE Trans. Softw. Eng. **29**(8), 687–704 (2003). doi: 10.1109/TSE.2003.1223644
169. Tichy, W.F.: Should computer scientists experiment more? IEEE Comput. **31**(5), 32–39 (1998)
170. Tichy, W.F., Lukowicz, P., Prechelt, L., Heinz, E.A.: Experimental evaluation in computer science: a quantitative study. J. Syst. Softw. **28**(1), 9–18 (1995)
171. Trochim, W.M.K.: The Research Methods Knowledge Base, 2nd edn. Cornell Custom Publishing, Cornell University, Ithaca (1999)
172. van Solingen, R., Berghout, E.: The Goal/Question/Metric Method: A Practical Guide for Quality Improvement and Software Development. McGraw-Hill International, London/Chicago (1999)
173. Verner, J.M., Sampson, J., Tosic, V., Abu Bakar, N.A., Kitchenham, B.A.: Guidelines for industrially-based multiple case studies in software engineering. In: Third International Conference on Research Challenges in Information Science, Fez, pp. 313–324 (2009)
174. Vinson, N.G., Singer, J.: A practical guide to ethical research involving humans. In: Shull, F., Singer, J., Sjøberg, D. (eds.) Guide to Advanced Empirical Software Engineering. Springer, London (2008)
175. Votta, L.G.: Does every inspection need a meeting? In: Proceedings of the ACM SIGSOFT Symposium on Foundations of Software Engineering, ACM Software Engineering Notes, vol. 18, pp. 107–114. ACM Press, New York (1993)
176. Wallace, C., Cook, C., Summet, J., Burnett, M.: Human centric computing languages and environments. In: Proceedings of Symposia on Human Centric Computing Languages and Environments, Arlington, pp. 63–65 (2002)

177. Wohlin, C., Gustavsson, A., Höst, M., Mattsson, C.: A framework for technology introduction in software organizations. In: Proceedings of the Conference on Software Process Improvement, Brighton, pp. 167–176 (1996)
178. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering: An Introduction. Kluwer, Boston (2000)
179. Wohlin, C., Aurum, A., Angelis, L., Phillips, L., Dittrich, Y., Gorschek, T., Grahn, H., Henningsson, K., Kågström, S., Low, G., Rovegård, P., Tomaszewski, P., van Toorn, C., Winter, J.: Success factors powering industry-academia collaboration in software research. IEEE Softw. (PrePrints) (2011). doi: 10.1109/MS.2011.92
180. Yin, R.K.: Case Study Research Design and Methods, 4th edn. Sage Publications, Beverly Hills (2009)
181. Zelkowitz, M.V., Wallace, D.R.: Experimental models for validating technology. IEEE Comput. **31**(5), 23–31 (1998)
182. Zendler, A.: A preliminary software engineering theory as investigated by published experiments. Empir. Softw. Eng. **6**, 161–180 (2001). doi: http://dx.doi.org/10.1023/A:1011489321999

索　引

索引中所标页码为英文版原书页码，与页边栏中的页码一致。